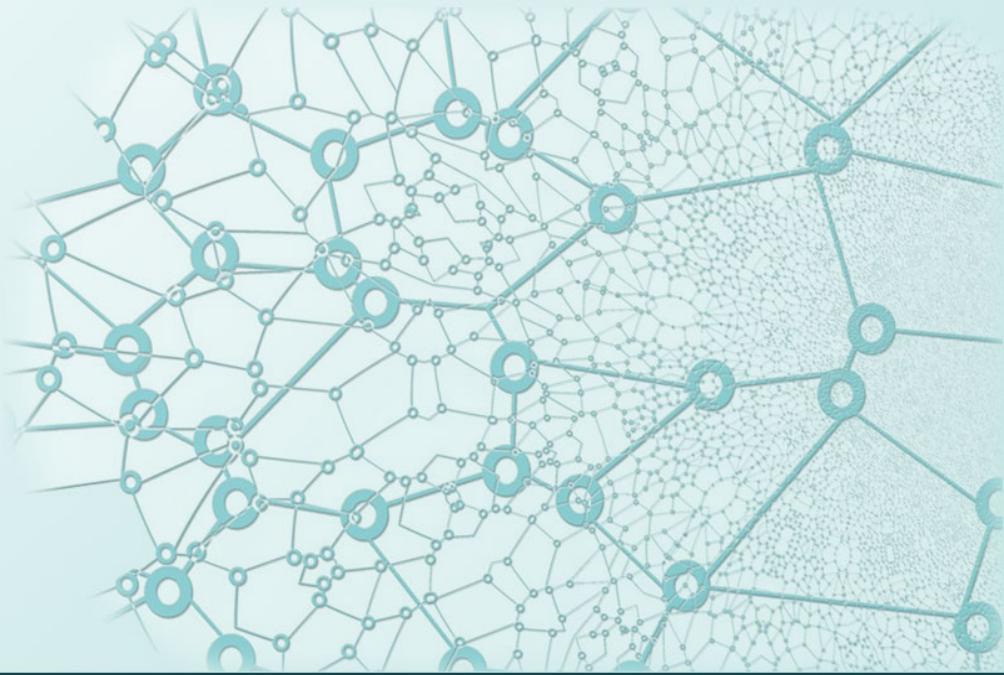
KNOM Review | 25권 2호, 2022년 12월 | ISSN 2287-1543 (Online)

KNOM REVIEW

Network Operations & Management

Vol. 25 No.2 | DEC. 2022



한국통신학회 통신망운용관리연구회

The Committee on Korean Network Operations and Management (KNOM)
Korean Istitute of Communication Sciences (KICS)
http://www.knom.or.kr

목 차

제25권 제2호, 2022년 12월

IBN@TEIN: QoS Assurance를 통한 서비스 구축을 위한 AI 기반 의도 기반 네트워킹 플랫폼
..... Mir Muhammad Suleman Sarwar, Sajid Alam, Afaq Muhammad, 송왕철 1

딥러닝 기반 네트워크 공격 및 침입 탐지 방법 연구
..... 홍지범, 최민지, 유재형, 홍원기 10

분할 NFT 기술에 대한 연구와 활용 방안 분석
..... 최원석, 우종수, 홍원기 22

GPU 데이터 캐시 접근 패턴 방법에 따른 성능 변화 분석
..... Theodora Adufu, 김윤희 30

경량화 데이터와 딥러닝 모델을 적용한 효율적인 네트워크 트래픽 분류 방법
..... 신창의, 박지태, 백의준, 최정우, 김명섭 40

네트워크 트래픽 분석을 통한 규칙 기반 사용자 행위 탐지 시스템 설계
..... 박지태, 신창의, 백의준, 최정우, 김명섭 51

한국통신학회 통신망운용관리연구회

https://doi.org/10.22670/knom.2022.25.2.1

IBN@TEIN: QoS Assurance를 통한 서비스 구축을 위한 AI 기반 의도 기반 네트워킹 플랫폼

미르 무함마드 술래만 사르워◆, 사지드 알람*, 아팍 모하마드**, 송 왕 철°

IBN@TEIN: An AI-driven Intent-based Networking Platform for Service Deployment with QoS Assurance

Mir Muhammad Suleman Sarwar◆, Sajid Alam*, Afaq Muhammad**, Wang-Cheol Song°

요 약

본 논문은 QoS (Quality of Service) 매개 변수를 사용한 서비스 보장을 위한 플랫폼을 제시한다. 서비스는 여러 OpenStack 기반 클라우드 인스턴스에서 호스팅된다. 서비스는 VXLAN 또는 GENEVE 터널링과 같은 오버레이 네트워크를 사용하여 클라우드 인스턴스 간에 공유된다. 오버레이 네트워크는 IBN(Intent-based Networking Application)에 의해 배포되고 관리된다. 소스와 목적지 사이의 통신은 SDN-Controller를 사용하여 경로를 선택하여 머신 러닝(ML)으로 지능적으로 제어된다. IBN은 한국, 파키스탄, 캄보디아, 말레이시아의 TEIN (Trans-Eurasia Information Networking)에 배치된다. 비디오 서비스와 같은 서비스는 HD, UHD 또는 SD와 같은 서로 다른 QoS 매개 변수를 가진 클라우드 인스턴스에서 호스팅된다. 사용자는 서비스 요청에 대해 추상 형식으로 의도를 제출한다. IBN은 Intents를 필요한 네트워크 구성으로 변환하고 Intent Assurance를 위해 기본 인프라에 구성을 배포한다. 클라우드가 제공하는 서비스를 다른 클라우드와 공유해야 지연 시간 및 단일 장애 지점을 줄일 수 있다.

Key Words : IBN, TEIN, GENEVE tunneling, OpenStack, Machine Learning

ABSTRACT

This paper presents a platform for service assurance with Quality of Service (QoS) parameters. The services are hosted at multiple OpenStack-based Cloud Instances. Services are shared between Instances of Clouds using an Overlay network such as VXLAN or GENEVE tunneling. The overlay network is deployed and managed by an Intent-based Networking Application (IBN). Communication between source and destination is intelligently controlled with Machine Learning (ML) by selecting paths using SDN-Controller. IBN is deployed at Trans-Eurasia Information Networking (TEIN) in South Korea, Pakistan, Cambodia, and Malaysia. Services such as video services are hosted at Instances of Clouds with different QoS parameters such as HD, UHD, or SD. A user submits an Intent in abstract form for a service request. IBN translates Intents into necessary network configurations and deploys configurations over the underlying infrastructure for Intent Assurance. Services provided by Cloud sometimes need to be shared with other Clouds to reduce latency and single points of failure.

This work was supported by the 2022 education, research and student guidance grant funded by Jeju National University.
◆ First Author : Jeju National University, Department of Computer Engineering, mirmohdsuleman@gmail.com
° Corresponding Author : Jeju National University, Department of Computer Engineering, philo@jejunu.ac.kr
* Jeju National University, Department of Electrical Engineering, sajid.alam19801@gmail.com
** Jeju National University, Department of Computer Engineering, afaq24@gmail.com
논문번호 : KNOM2022-02-01, Received November 16, 2022; Revised December 10, 2022; Accepted December 16, 2022

Ⅰ. INTRODUCTION

One of the key capabilities of IBN is the ability to monitor, identify and react in real time to changing network conditions. This pertains to network (and related applications and services) provisioning and administration, as well as proactively and autonomously managing change in operational conditions.

To this end, in this paper, we apply IBN on TEIN [1] to automate service deployment with QoS assurance. It includes the computation of the best path between nodes deployed at partner countries, i.e., Korea, Pakistan, Malaysia, and Cambodia, over the high-speed TEIN. The testbed consists of 1) the IBN system, which is a closed-loop application that automatically handles the generation of policies to control the network, 2) Clouds deployed at each country that host services, 3) a real-time monitoring system that collects the overall network statistics, 4) an ML-driven system that predicts link utilization and computes the best path based on network statistics provided by the monitoring system.

We propose intelligent communication between Service requesting nodes (SRN) and Service Providing nodes (SPN) in partner countries. The IBN system, including an intelligent J-box, will be deployed at the Node of Jeju National University, whereas one intelligent J-box will be deployed at each Node of a partner country, as mentioned in Table 1.

Table 1. List of Nodes and their respective Equipment Details

No.	Nodes	Equipment to be Deployed	Country
1	Jeju National University	Deployment of 1x Intelligent J-box	South Korea
2	KOREN-NOC	Deployment of 1x Intelligent J-box	South Korea
3	University of Malaya	Deployment of 1x Intelligent J-box	Malaysia
4	Institute of Technology of Cambodia (ITC)	Deployment of 1x Intelligent J-box	Cambodia
5	Pakistan Education & Research Network (PERN)	Deployment of 1x Intelligent J-box	Pakistan
6	National Univ. of Science & Tech. (NUST)	Deployment of 1x Intelligent J-box	Pakistan
7	Sarhad University of Science & IT (SUIT)	Deployment of 1x Intelligent J-box	Pakistan

Ⅱ. RESEARCH BACKGROUND

Internet Engineering Task Force (IETF) defines an Intent-Based Network (IBN) as "a network that can be managed using intent" [2]. An Intent is "a set of operational goals (that a network should meet) and outcomes (that a network is supposed to deliver), defined in a declarative manner without specifying how to achieve or implement them" [2]. In an IBN, a user provides a networking requirement in an abstract manner that does not contain any technical configurations. An abstract networking requirement is called an Intent. The IBN application itself decides the necessary configurations for satisfying an Intent. The IBN application executes commands on the underlying network for deploying the decided configurations. This process is called intent assurance. An Intent can be any abstract network requirement, e.g., a request for a video service for which some network configurations are required, such as network slicing. Another example of Intent is "deploy tunnel from node A to node B." It is then the job of the IBN application to translate the Intent into network configurations based on some policy. Different IBNs are developed for assurance of different types of Intent. A study proposed network slicing in 5G networks using Generative Adversarial Neural Network (GAN) and IBN [3]. Another study presented a generic Intent-based networking platform for E2E network slice orchestration and

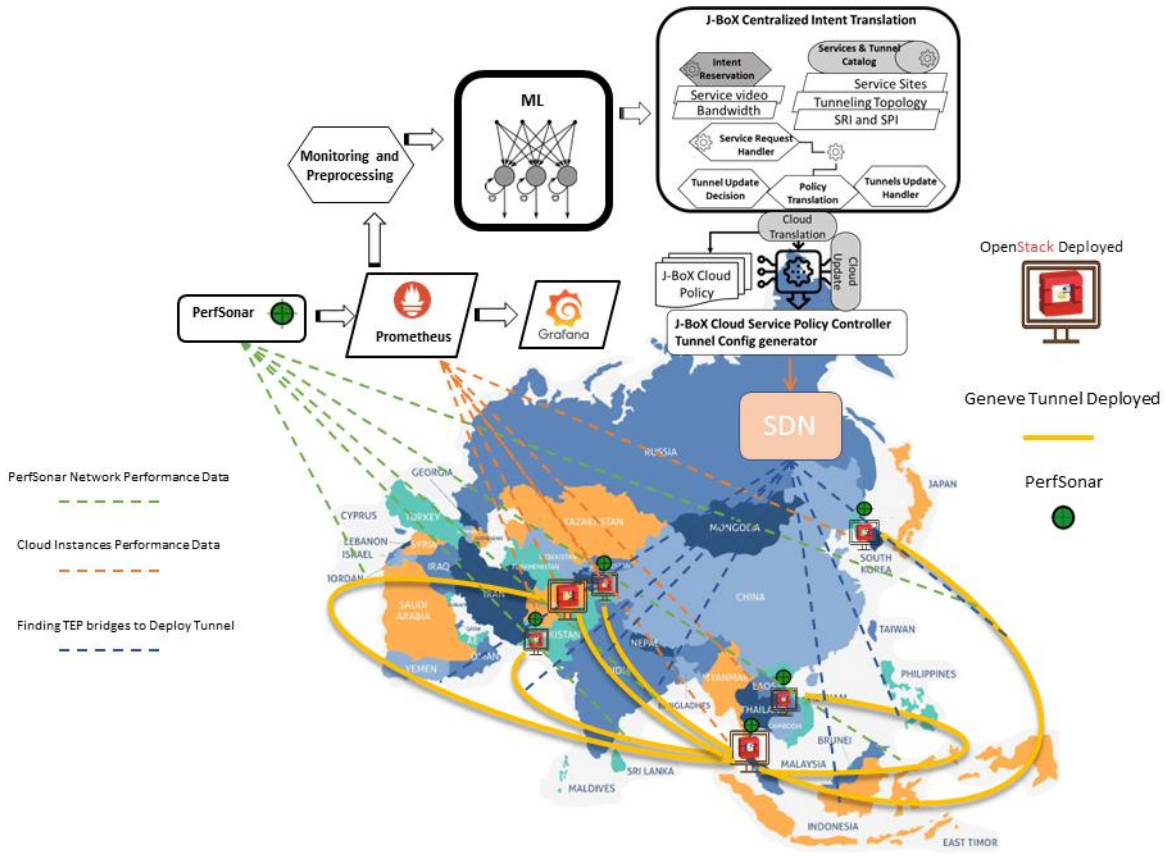


Fig. 1. IBN@TEIN: QoS Assurance를 통한 서비스 구축을 위한 AI 기반 의도 기반 네트워킹 플랫폼

lifecycle management using IBN and GNN (Graph neural Network) [4]. Zeydan, Engin, and Yekta Turk, in a study, provide a comprehensive survey on IBN [5]

To leverage the advantages of virtualization Data Centers have moved to Cloud [6]. Cloud provides shared resources as a Service[7], such as Platform as a Service (PaaS), Software as a Service (SaaS), and Storage as a Service (Storage as a Service)[8]. Cloud services sometimes need to be shared with other Clouds to reduce latency and single points of failure [9]. Cloud can only communicate with another Cloud using an overlay network such as VXLAN [10] or GENEVE tunneling [11]. The overlay network between Clouds must be managed for efficient communication between Clouds. For this reason, ML models can forecast the performance of the network and Cloud. IBN takes network management decisions such as path selection based on ML results and provides input to an SDN Controller. Clouds communicate over the path selected by IBN.

Ⅲ. SYSTEM DESIGN

Fig. 1 shows the placement of nodes in partner countries. At each Node, we can see OpenStack Cloud that hosts services. Monitoring tools such as PerfSonar and Node Exporter provide data to Prometheus, a time series database (TSDB). A Machine Learning Model provides input to the IBN based on data obtained from monitoring tools.

Fig. 1 shows a virtual overlay network topology that provides network connectivity to the OpenStack-based clouds at each site of a partner country. We can see that IBN learns about the network traffic on paths and Cloud Nodes. IBN deploys all tunnels and decides an optimal route for each destination. OpenStack-based Cloud is deployed at geographically diverse locations. We

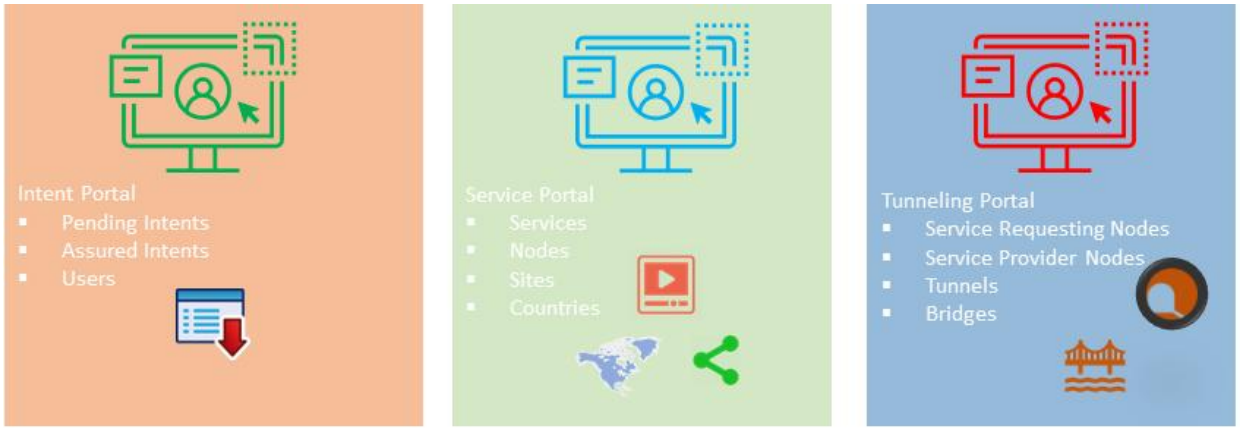


Fig. 2. IBN 웹 포털

have data or files at each site, such as video services hosted at OpenStack nodes. This IBN-based platform consists of the following main components:

3.1 IBN
3.2 OpenStack-based Clouds
3.3 Overlay Network
3.4 SDN Controller
3.5 Monitoring tools
3.6 Machine Learning

3.1 IBN

The IBN application contains complete system information and has a global view of the current system status. The IBN application receives an Intent, such as acquiring a service. With ML input IBN learns the most optimal path to a target node.

Once the overlay network sets the connectivity, a service is provided by tunneling from source to destination instances of Clouds. As shown in Fig. 2, the IBN Portal is used to submit user intents. The IBN portal also contains information on Services hosted at Cloud Sites and L2 tunnels that form the overlay topology.

Initially, Intents are pending. The IBN Portal shows all pending and assured Intents. The IBN Portal shows each Service at each Node and site of a country. It also provides interfaces for adding new services. The IBN Portal shows every tunnel between a source and a destination Cloud. This portal also shows tunneling details, such as bridges, ports, and remote IPs between SRN and SPN, as shown in Fig. 2. L2 tunnels are unidirectional, so for each SRN and SPN, two tunnels are deployed for bi-directional communications.

In addition, the IBN application ensures the provision of a requested service to a user. A user from any node from any partner country can request Service at any other node in any other partner country. IBN accepts user intent and creates then deploys network configuration per user Intent. User intent is a requirement, in simple words, for certain complex network configurations. For example, an intent can be a user request for a service such as a video or a VM with OpenStack. The IBN application has a Catalog that contains information on Nodes, Links between Nodes, Services at each Node, and resources of each Node. The requested Service can be a video or a VM with an OpenStack deployment. The requested video can have different qualities, such as HD or UHD. Nodes that provide services have computing resources such as CPU, Memory, and Storage.

3.2 OpenStack-based Clouds

Fig. 1 also shows tunnels created between nodes. Note that any node can be an SRN or SPN. Sometimes a node can be both SRN and SPN in a scenario where the Node hosts a service but also requires a service from another node. Every Node at each partner country is an

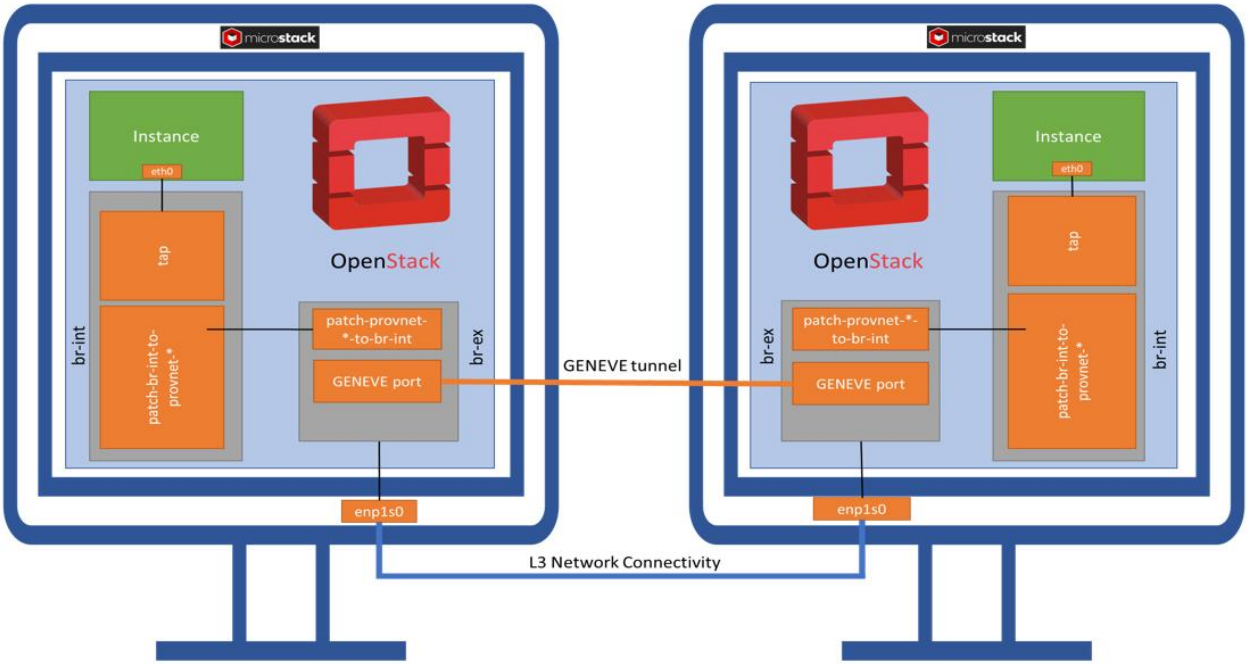


Fig. 3. OpenStack 클라우드 간 GENEVE 터널링

OpenStack-based [12] node, either a Control node or Compute Node. This way, we have an OpenStack Cloud in geographically diverse locations. Services are hosted on instances at OpenStack Compute nodes. Each Node also has the Intelligent J-box deployed. An Intelligent J-box has an OpenStack Cloud and monitoring tools.

3.3 Overlay Network

Communication between Clouds is achieved using an overlay network. The overlay network is established by deploying L2 tunnels such as VXLAN or GENEVE tunnels using default bridges of Cloud's Open Flow Virtual Switches such as OVS or OVN[13]. An L2 tunnel is unidirectional, and the tunnel must be deployed from both directions. Fig. 3 shows a GENEVE tunnel deployed on the default bridge br-ex of OpenStack Cloud.

3.4 SDN Controller

Communication between a source and a destination Cloud can take several paths. An SDN controller such as RYU [14] connected to each OVS of the Cloud can control traffic over a selected path. The chosen path is provided to the SDN-Controller by the IBN.

3.5 Monitoring Modules

The Monitoring module will monitor all the network links and the Node's CPU, Memory, and Storage in real-time and provide input to the ML module. A times series database (TSDB) such as Prometheus [15] stores all data periodically.

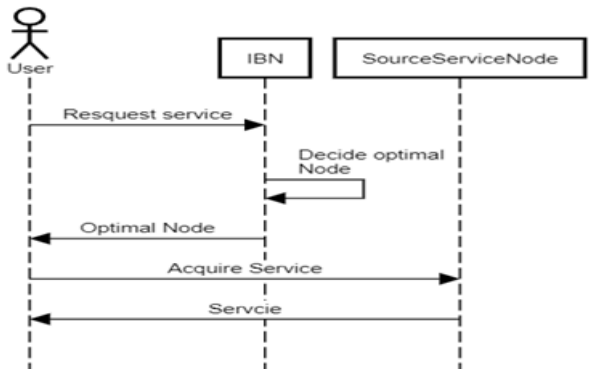


Fig. 4. 서비스 사용자 요청 워크플로우

Grafana [16] is then used to visualize Prometheus data. This data is then used for training Machine Learning models.

3.6 Machine Learning

The ML module trains on data and predicts

future resource utilization of each Node and the best-forecasted path. IBN application is updated with each Node's total, utilized, and predicted utilization of resources. For monitoring, Node Exporter provides data fetched by Prometheus, whereas Grafana is used for data visualization.

Two general types of ML methods can be utilized for processing Cloud resources and link data. Traditional ML algorithms such as K-Nearest Neighbors, Support Vector Machines, and Random Forests involve complex feature engineering. Recent ML algorithms or deep learning algorithms such as CNN and LSTM are based on end-to-end learning with an intrinsic feature engineering mechanism. The latter has greater learning capabilities, provided it is fed with more training data.

A further categorization of ML algorithms in the context of resource orchestration by their optimization objectives is as follows. ML-based regression models can predict continuous output by studying the relationship between production and one or more input variables. To this end, it can uncover the relations between different

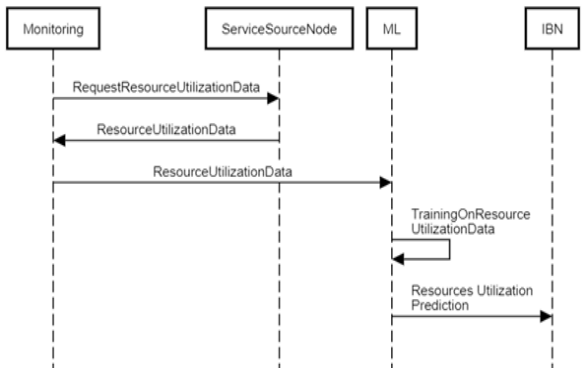


Fig. 5. IBN에 대한 ML 입력 워크플로우

performance metrics. Classification algorithms label data into different classes. They can be useful in identifying key application components or abnormal system behavior. Decision models can be employed for decision-making in resource provisioning by resolving the decision-making process in identifying choices with maximized cumulative rewards. Popular algorithms such as model-free and model-based reinforcement learning fall into this category. Time-series analysis techniques are a useful ML tool for behavior modeling of sequential data, including workload arrival rates or resource utilization.

Ⅳ. Workflow

As depicted in the sequence diagrams of Fig. 4 and Fig. 5, a user requests the IBN application for a service such as a VM with OpenStack deployed. The IBN application decides the most optimal Node for the requested Service. An optimal node is a Node that provides the requested Service with optimal resource utilization. After that, the IBN application creates the necessary network configurations to provide this Service, such as tunneling between source to destination Cloud or communication over a selected path. In the end, the IBN application deploys these network configurations on the network.

The proposed system deploys network configurations to provide a service to a user based on a user Intent. The Service is provided to the user from the most optimal Node. An optimal node for a service request is decided based on ML predictions by the IBN application. This way, the proposed system intelligently assures a user intent.

Ⅴ. RESULTS

Fig. 6 shows the Horizon dashboard of two OpenStack Clouds at Malaysia and South Korea. Fig. 7 shows tunnel configuration of Cloud South Korea and Cloud Malaysia. Fig. 8 shows theoverlay network topology between all Clouds at RYU SDN Controller. Each of the links shown in the topology is a GENEVE tunnel. Each of the virtual switch shown in topology is an Open Virtual Switch of OpenSatck Cloud. The topology of Cloud Nodes and GENEVE tunnels can be made visible by setting the SDN-Controller as the RYU to the bridge "br-ex" on which

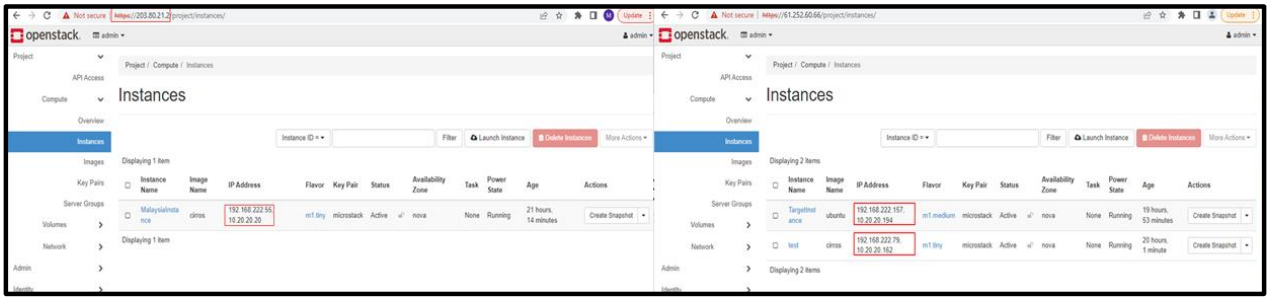


Fig. 6. Cloud Malaysia 및 Cloud South Korea Horizon 대시보드

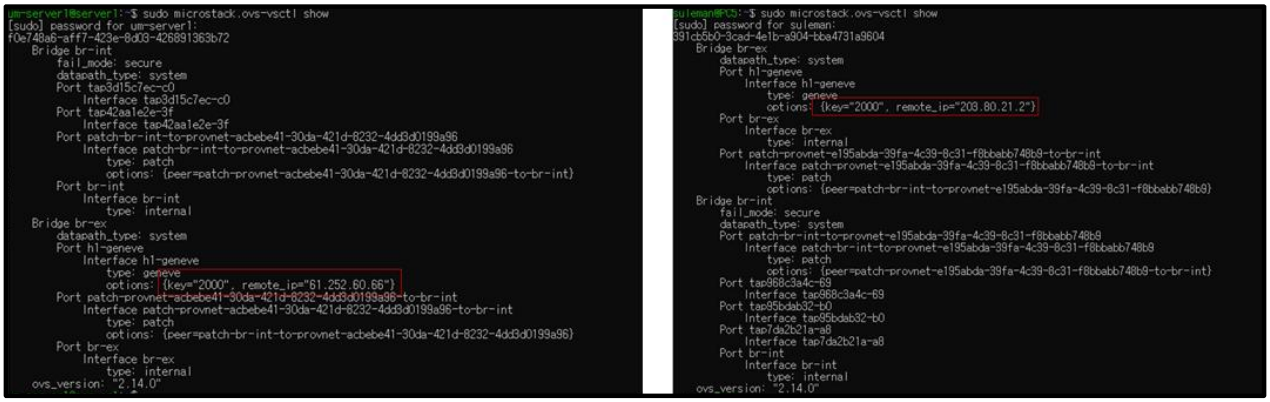


Fig. 7. 클라우드 말레이시아와 클라우드 대한민국 GENEVE 터널 구성

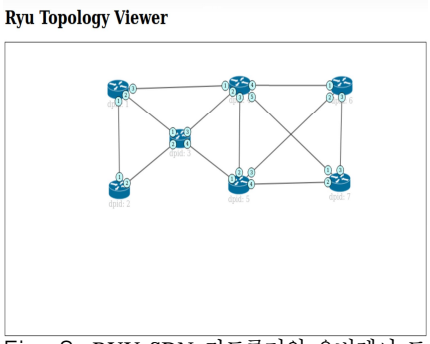


Fig. 8. RYU SDN 컨트롤러의 오버레이 토폴로지

GENEVE tunnel is deployed. Once the SDN-Controller is attached to the bridge br-ex SDN Controller visualizes the overlay network topology, port numbers on which GENEVE tunnels are deployed as well as flow rules deployed on each switch. The SDN-Controller can then control traffic between a source and destination by adding flow rules accordingly.

Fig. 9 and Fig. 10 finally show connectivity testing between an Instance of Cloud South Korea and an Instance of Cloud Malaysia. Fig. 9 shows an Instance in Cloud korea with IP 192.168.22.157, 10.20.20.194 can now access an instance in Cloud Malaysia with IP 192.168.222.55, 10.20.20.20. Fig. 10 shows that the Instance with IP 192.168.222.55, 10.20.20.20 in Cloud Malaysia can now access the Instance of Cloud Koreawith IP 192.168.222.157, 10.20.20.194

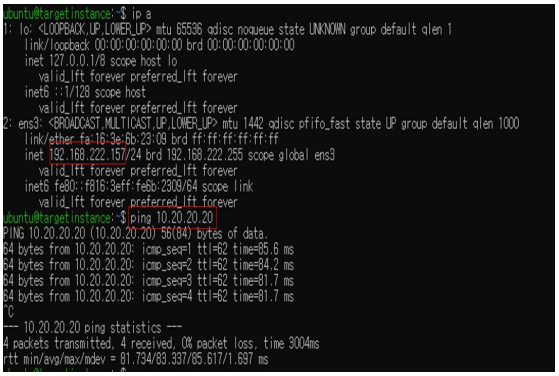


Fig. 9. 클라우드 인스턴스 대한민국에서 클라우드 말레이시아 인스턴스에 액세스

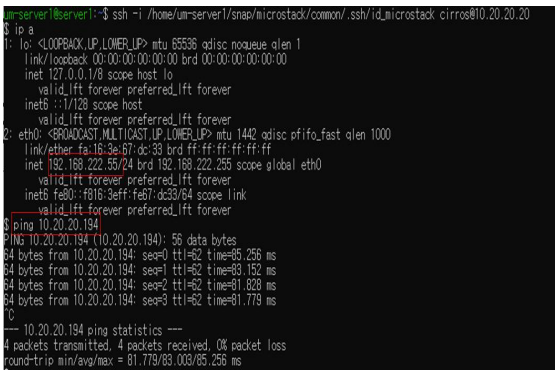


Fig. 10. 클라우드 말레이시아 인스턴스 액세스 대한민국

VI. CONCLUSION

We aim to provide the Intent-Based Networking (IBN) solution for provisioning multiple networking, compute processing, and data sharing services to perform research and development. As there are multiple service provisioning goals and the utilized infrastructure is highly-distributed where each Service has its requirements, it is highly required to manage the network using an advanced and simplified solution. Hence, we propose to develop and implement a solution through which we can orchestrate and control services across highly distributed TEIN and NRENs infrastructure. IBN system exposes a simplified front-end web-based interface for a user to request for the automated orchestration of services across multi-site globally distributed infrastructure. IBN uses the user's high-level service context/ business goals and translates them into low-level system configurations.

References

[1] Asia@Connect: https://www.tein.asia/

[2] A. C. Clemm, L Granville, L Tantsura, J. (2019). Intent-Based Networking-Concepts and Overview. Available: https://bit.ly/2ImukBF

[3] K. A. Abbas, Muhammad Khan, Talha Ahmed Mehmood, Asif Song, Wang-Cheol, "IBNSlicing: intent-based network slicing framework for 5G networks using deep learning," in 2020 21st Asia-Pacific Network Operations and Management Symposium (APNOMS), 2020, pp. 19-24: IEEE.

[4] T. A. Khan, K. Abbass, A. Rafique, A. Muhammad, and W.-C. Song, "Generic intent-based networking platform for E2E network slice orchestration & lifecycle management," in 2020 21st Asia-Pacific Network Operations and Management Symposium (APNOMS), 2020, pp. 49-54: IEEE.

[5] E. Zeydan and Y. Turk, "Recent advances in intent-based networking: A survey," in 2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring), 2020, pp. 1-5: IEEE.

[6] P. Mell and T. Grance, "The NIST definition of cloud computing," 2011.

[7] R. Chayapathi, S. F. Hassan, and P. Shah, Network Functions Virtualization (NFV) with a Touch of SDN: Netw Fun Vir (NFV EPub_1. Addison-Wesley Professional, 2016.

[8] L. Wang, R. Ranjan, J. Chen, and B. Benatallah, Cloud computing: methodology, systems, and applications. CRC Press, 2017.

[9] S. Gopinath and E. J. I. J. A. E. R. Sherly, "A comprehensive survey on data replication techniques in cloud storage systems," vol. 13, no. 22, pp. 15926-15932, 2018.

[10] M. Mahalingam et al., "Virtual extensible local area network (VXLAN): A framework for overlaying virtualized layer 2 networks over layer 3 networks," 2070-1721, 2014.

[11] J. Gross, I. Ganga, and T. J. I. d. Sridhar, "Geneve: Generic network virtualization encapsulation," 2014.

[12] Openstack: https://www.openstack.org/

[13] M. M. S. Sarwar, J. J. D. Rivera, A. Muhammad, and W.-C. Song, "GENEVE@ TEIN: A Sophisticated Tunneling Technique for Communication between OpenStack-based Multiple Clouds at TEIN," in 2022 23rd Asia-Pacific Network Operations and Management Symposium (APNOMS), 2022, pp. 1-4: IEEE.

[14] Ryu SDN: https://ryu.readthedocs.io/en/latest/

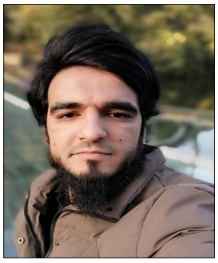
[15] Prometheus: https://prometheus.io/

[16] R. McCollam, "Grafana Enterprise," in Getting Started with Grafana: Springer, 2022, pp. 339-354.

미르 무함마드 술래만 사르워 (Mir Muhammad Suleman Sarwar)

Mir Muhammad Suleman Sarwar received his BS degree in Information Technology from Agricultural University Peshawar in 2010. He received his MS degree in Computer Sciences from University of Peshawar, Pakistan in 2015. In 2021, he joined Network Convergence Lab as a PhD scholar. His research interests include the Overlay Networks, Cloud, SDN, NFV, Intent Based Networking, machine learning and VNF development.

사지드 알람 (Sajid Alam)



Sajid Alam received his BS degree in Electrical Engineering from National University of FAST karachi Pakistan, MS degree in Electrical Engineering from University of North China Electric Power Beijing ,China. In 2022, he joins Network Convergence Lab as a PhD scholar. His research interests include the Overlay Networks, Cloud, SDN, Intent Based Networking and machine learning.

아팍 모하마드 (Afaq Muhammad)

Muhammad Afaq received a Ph.D. degree in Computer Engineering from Jeju National University, MS degree in Electrical Engineering with emphasis on Telecom from Blekinge Institute of Technology, Sweden, and BS degree in Electrical Engineering from the University of Eng. and Technology, Peshawar, Pakistan in 2017, 2010, and 2007 respectively. He is currently working as a Postdoctoral Researcher at Network Convergence Lab, Jeju National University. He also worked as an Assistant Professor in the Department of Computer Science and IT at the Sarhad University of Science and IT, Pakistan. Before starting his Ph.D., he worked as a Research Associate in the Faculty of Computer Science and Engineering at GIK Institute of Engineering Sciences and Technology, Pakistan, and as a Lecturer in the Department of Electrical Engineering at the City University of Science and IT. His research interests are cloud computing, software-defined networking, network function virtualization, wireless networks, and protocols, machine learning, and data science.

송 왕 철 (Wang-Cheol Song)

Wang-Cheol Song has been working as a Professor at the Computer Engineering Department, Jeju National University, South Korea, since 1996. He received B.S. degree in Food Engineering and Electronics from Yonsei university, Seoul, Korea, in 1986 and 1989, respectively. And He received M.S. and Ph.D. in Electronics studies from Yonsei University, Seoul, Korea, in 1991 and 1995, respectively. His research interests include VANETs and MANETs, SDN/NFV, Intent-based networking, and network management.

KNOM Review '22-02 Vol.25 No.02
https://doi.org/10.22670/knom.2022.25.2.10

딥러닝 기반 네트워크 공격 및 침입 탐지 방법 연구

홍 지 범◆, 최 민 지**, 유 재 형*, 홍 원 기°

A Study of Network Attack and Intrusion Detection Method using Deep Learning

Jibum Hong◆, Minji Choi**, Jae-Hyoung Yoo*, James Won-Ki Hong°

요 약

안정적인 네트워크 관리를 위해서는 DoS/DDoS와 같은 네트워크 공격이나 침입과 같은 비정상적인 트래픽을 초기에 탐지하고 예방하는 것이 중요하다. 본 논문에서는 Recurrent Neural Network (RNN) 및 Transformer와 같은 딥러닝 모델을 기반으로 네트워크 공격 및 침입 트래픽을 탐지하는 방법을 제안한다. 그리고 제안하는 방법을 실제 네트워크 환경에 적용하기 위한 시스템을 설계한다. 비정상 트래픽의 분류 (탐지) 성능을 향상시키기 위해, 제안하는 모델은 이전 시간의 예측 결과를 활용하여 이전 시간 인덱스의 출력을 새 입력에 제공한다. 제안하는 모델의 성능 검증은 서로 다른 features를 가지는 세 가지 네트워크 공격 관련 공개 데이터셋 (NSL-KDD, UNSW-NB15, CICIDS 2017)을 사용하여 이진 (binary) 및 다중 클래스 (multi-class) 분류를 통해 그 성능을 검증한다. 실험 결과, 제안하는 모델은 3개의 데이터 세트에 대해 이진 분류에서 각각 0.956, 0.938 및 0.997의 개선된 성능 (F1 score)를 보여준다.

Key Words : Network Attack&Intrusion Detection, Deep Learning, Network Management

ABSTRACT

For stable network management, it is important to detect and prevent network attacks (e.g., DoS/DDoS) or abnormal traffic at an early stage. In this paper, we propose a method for detecting network attacks and intrusion traffic based on a deep learning model which has a sequential structure such as Recurrent Neural Network (RNN) and Transformer. To improve the classification performance, our proposed model uses the previous prediction to feed the output of the previous time index to new input. And we propose a system design to apply the proposed method to the real-world network environment. We evaluate our method through binary and multi-class classifications by using three public datasets (NSL-KDD, UNSW-NB15, CICIDS 2017) which have different features. Our model shows the improved F1 score 0.956, 0.938, and 0.997 in binary classification for the three datasets, respectively.

※ 이 논문은 2022년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원 (2018-0-00749, 인공지능 기반 가상 네트워크 관리기술 개발)과 대학ICT연구센터육성지원사업의 연구결과로 수행되었음 (IITP-2022-2017-0-01633).
◆ First Author : Pohang University of Science and Technology (POSTECH), Department of Computer Science Engineering, hosewq@postech.ac.kr
° Corresponding Author : POSTECH, Department of Computer Science Engineering, jwkhong@postech.ac.kr
* POSTECH, Department of Computer Science Engineering, jhyoo78@postech.ac.kr
** POSTECH, Graduate School of Artificial Intelligence, minjichoi@postech.ac.kr
논문번호 : KNOM2022-02-02, Received November 30, 2022; Revised December 10, 2022; Accepted December 16, 2022

Ⅰ. 서 론

네트워크가 발전하고 다양한 서비스가 등장하면서 트래픽이 급격하게 증가하였으며, 이로 인해 정상 트래픽과 더불어 네트워크 공격이나 침입 등의 비정상 트래픽 또한 증가하고 있다. 이러한 비정상 트래픽은 DoS/DDoS와 같은 서비스 거부 공격과 또는 공격을 준비하기 위한 port scan과 같은 스캔 공격 등 다양한 원인에 의해 발생하며, 이는 서버의 정상적인 운용과 네트워크 기능에 심각한 영향을 미친다 [1].

비정상 트래픽 분류 및 네트워크 공격 탐지를 위한 기존 방법들은 잘 알려진 포트 번호를 이용한 포트 기반 탐지 방법 또는 트래픽 분류를 위한 IP 패킷의 페이로드에 포함된 시그니처를 이용한 페이로드 기반 탐지 방법이 존재한다. 그러나 기존 방법들은 네트워크 공격 방법이 다양화됨에 따라 현재 그 한계가 존재한다 [2].

이러한 문제를 해결하기 위해 네트워크 트래픽 통계 정보를 기반으로 그 특징을 분석하여 비정상 트래픽을 탐지하는 통계 기반 방법이 제안되었다. 특히, 최근에는 인공지능 (Artificial Intelligence, AI)과 머신러닝 (Machine Learning, ML)의 발전에 따라 통계 기반 분석 방법에 머신러닝/딥러닝 (Deep Learning, DL)을 사용하여 네트워크 공격 및 침입 트래픽을 탐지하는 방법이 주목받고 있다 [3].

본 논문은 네트워크 보안 기능에 초점을 맞추어 지도학습을 기반으로 하는 딥러닝 기반 네트워크 공격 및 침입 탐지 방법을 제안한다. 제안하는 방법은 Recurrent Neural Network (RNN)과 Transformer 등의 딥러닝 알고리즘을 활용한다. 또한, 제안하는 방법은 자연언어 처리 (Natural Language Processing, NLP)에서 언어 모델 학습에 사용되는 이전 시간 인덱스의 예측 결과를 활용하는 방법 (previous prediction) [4]을 적용하여 탐지 성능 개선한다. 마지막으로, 제안하는 탐지 모델을 실제 네트워크 환경에 적용하기 위해 네트워크 공격 및 침입 탐지 시스템을 설계한다. 성능 검증은 네트워크 공격과 관련하여 잘 알려진 공개 데이터 셋들 (NSL-KDD [5], UNSW-NB15 [6], CICIDS 2017 [7])을 이용하여 다른 머신러닝/딥러닝 기반 모델들과 탐지 성능을 비교 및 분석하여 그 성능을 검증하고자 한다.

본 논문의 주요 기여 내용은 다음과 같다.

- 개선된 성능의 이진 분류를 통해 비정상 트래픽을 탐지하고, 다중 클래스 분류를 통해 비정상 트래픽을 분석하여 네트워크 공격 및 침입 유형 분류
- 머신러닝/딥러닝 알고리즘들과 관련된 네트워크 공격 및 침입 탐지 방법 및 최신 연구들에 대한 조사 및 요약
- 시계열 특성을 가지는 트래픽 데이터 분류 성능 향상을 위해 이전 시간 인덱스 예측 결과를 활용하는 방법 (previous prediction 기능) [4] 제안
- 제안된 방법을 활용하여 폐쇄 루프 기반 네트워크 자동화 측면에서의 네트워크 공격 및 침입 탐지 시스템 설계

본 논문의 2장에서는 제안하는 탐지 방법과 관련된 배경 지식을 제공하고 관련 연구를 논의한다. 3장에서는 본 논문에서 제안하는 탐지 방법과 모델을 기반으로 하는 탐지 시스템을 설계한다. 이후 4장에서는 제안하는 방법을 기반으로 구현된 모델의 성능을 검증한다. 마지막으로, 5장에서는 결론 및 향후 연구를 서술한다.

Ⅱ. 연구 배경 및 관련 연구

2.1. 네트워크 공격 및 침입 탐지

네트워크 공격 및 침입 (Attack&Intrusion)은 IDS (Intrusion Detection System)와 같은 보안 절차 혹은 시스템/서비스의 취약점을 악용하여 시스템/서비스의 보안 정책을 위반하거나 피해를 야기하는 일련의 행동으로 정의할 수 있다 [8].

대표적인 네트워크 공격 및 침입 탐지 시스템인 Network Intrusion Detection System (NIDS)은 네트워크 트래픽을 모니터링하고 분석하여 인가되지 않은 행위 또는 악의적인 행위를 식별하고 공격 및 침입을 실시간으로 탐지하여 네트워크 관리자에게 보고한다. 이와 같이 비정상적인 트래픽을 모니터링하고 분석하는 기존 방법들은 트래픽의 페이로드 (payload)에 포함된 포트 번호나 시그니처 정보를 사용한다. 하지만, 기존 방법들은 네트워크 공격과 침입이 더욱 은밀하고 다양해짐에 따라 그 한계를 지닌다. 이를 해결하기 위해 트래픽의 통계 정보를 활용하여 네트워크 트래픽의 행동이나 특성을 분석하는 통계 기반 분석 방법이 등장하였다.

네트워크 공격 및 침입 탐지 방법은 현재 1) 시그니처 기반 탐지 방법, 2) 통계 기반 탐지 방법 [9] 크게 두 가지 접근 방식으로 분류할 수 있다.

시그니처 기반 탐지 방법은 공격의 특징을 나타내는 시그니처 정보를 탐지 시스템에 미리 등록하고, 트래픽과의 패턴매칭을 통해 네트워크 공격 및 침입을 탐지한다. 반면, 통계 기반 탐지 방법은 트래픽으로 인한 시스템의 동작이나 트래픽 통계 정보를 포함하는 메트릭 값이 규칙적이거나 주기적 패턴을 벗어날 때 해당 네트워크 트래픽을 공격 및 침입으로 간주한다 [9], [10].

본 논문은 통계 기반 탐지 방법을 딥러닝을 기반으로 학습 및 분석하여 공격 및 침입 트래픽을 탐지한다. 이 방법은 딥러닝을 이용하여 데이터에서 이상치, outlier, 그리고 이상 상태 등의 예상하지 못한 패턴을 찾는다 [11]. 또한 네트워크 공격 및 침입과 같은 비정상 상태를 나타내는 트래픽은 정상 상태를 나타내는 트래픽에 비해 그 양이 매우 적기 때문에, 공격 및 침입 탐지는 매우 많은 양의 정상 데이터에서 극소수의 비정상 트래픽을 감지하고 구별하는 것도 중요한 목표이다.

2.2. 시계열 정보 학습을 위한 딥러닝 모델

시계열 데이터와 같이 순차적 구조를 가지는 딥러닝 모델은 RNN뿐만 아니라 RNN의 변형 구조인 Bi-directional RNN (Bi-RNN)과 Transformer 모델을 포함한다. RNN은 시계열 데이터와 같은 순차적 정보로 이루어진 데이터를 효과적으로 학습하도록 설계된 모델이다. 그러나 RNN은 입력 시퀀스를 시간순으로 배치하기 때문에 이전 패턴을 기준으로 출력이 수렴되는 경향이 있다는 한계가 있다.

이 문제를 해결하기 위해 Bi-RNN 모델이 제안되어 기존 순방향 학습에 역방향 학습을 추가하고, 이를 hidden layer에 추가하여 성능을 개선하였다 [12]. Transformer는 시퀀스 데이터의 표현을 계산할 수 있고, 명시적이고 반복적인 연결 없이 self-attention 매커니즘을 사용하는 딥러닝 모델이다. 이러한 방법은 주로 자연언어 처리와 컴퓨터 비전 등 순차적 패턴을 모델링하는 분야에서 주로 활용되었다. Sequential 모델에 대한 자세한 설명은 [13], [14]에서 확인할 수 있다.

[4]에서는 자연언어 처리에서 언어 모델의 성능을 향상시키기 위해 이전 시간 인덱스의 예측 결과를 활용하는 개념을 제안한다. [4]는 신경망을 사용하여 단어 시퀀스에서 feature 벡터를 표현하기 위해 "direct" 및 "circle" 구조를 사용한다. 특히 "circle" 구조는 각 시간 인덱스에서 다음 후보 단어에 대해 이전 시간 인덱스의 결과를 반복적으로 활용한다. 네트워크 트래픽 데이터는 단어의 시퀀스와 유사하게 시계열 특징을 지니기 때문에 본 연구에서는 제안하는 방법의 성능을 향상시키기 위해 RNN 계열의 모델 (RNN, Bi-RNN, Transformer)들을 활용하고, 이전 시간 인덱스에서 예측한 탐지 결과를 입력으로 다시 활용한다.

2.3. 관련 연구

최근 네트워크 공격 및 침입을 탐지 연구는 머신러닝/딥러닝 기반으로 연구가 많이 진행되고 있다 [15].

1) 머신러닝 기반 방법 : 머신러닝을 사용하여 네트워크의 공격 및 침입을 탐지하는 기존 연구는 Tree나 선형/비선형 모델 등을 많이 사용한다. 먼저, 최근에는 Decision Tree (DT) 계열의 모델들이 많이 연구되었으며, 기본 모델인 DT를 기반으로 XGBoost, GBM, Random Forest 등의 모델들도 비정상 트래픽 식별에 사용된다 [17, 18].

그리고 다른 기존 연구들은 선형 또는 비선형 모델을 사용하여 트래픽을 식별하는 Support Vector Machine (SVM)은 정확도를 높여 정상 및 비정상 트래픽을 예측하는데 사용된다 [19]. 또한, K-Nearest Neighbor (KNN)은 feature 간 유사성을 분석하여 특정 데이터 샘플의 클래스를 예측한다 [20, 21].

다음으로, K-Mean Clustering은 비지도학습 방식으로 데이터를 학습하는 반복적인 중심 기반 기계학습 알고리즘으로, 정상 트래픽과 비정상 트래픽의 클러스터들을 중심으로 트래픽 데이터와 각 클러스터의 중심 사이 거리를 계산하여 트래픽을 분류한다 [22].

2) 딥러닝 기반 방법 : 딥러닝은 머신러닝의 한 종류로 인공신경망을 통한 심층학습을 통해 데이터를 학습하는 방법이다. 먼저, Aritificial Neural Network (ANN)은 input layer, hidden layer, 그리고 output layer로 구성된 기본 모델로, 딥러닝을 이용한 탐지 연구 관련 초기 연구 [23]는 지도학습 및 비지도학습 데이터 모두를 사용하여 공격 및 침입을 탐지하기 위한 enhanced resilient back propagation artificial neural network (ERBP-ANN) 모델을 제안한다.

Deep Neural Network (DNN)은 복잡한 비선형 함수의 모델링에 적합하기 때문에 다양한 연구에서 사용된다. NIDS에 대해 공개적으로 사용 가능한 NSL-KDD, UNSW-NB15 및 CICIDS 2017 데이터

셋을 사용하는 DNN 기반 IDS 모델은 다른 ML 방법에 비해 향상된 결과를 보여준다 [24].

또한, RNN은 트래픽 데이터에서 시계열 특성을 학습하는데 일반적으로 흔히 사용되며, Long Short-Term Memory (LSTM)와 Gated Recurrent Unit (GRU)과 같은 RNN의 변형 모델 또한 공격 및 침입 트래픽 탐지에 활용된다 [25]. [26]의 경우, Multi-Layer Perceptron (MLP), softmax 분류기, GRU 모델을 사용하여 비정상 트래픽을 탐지하기 위한 RNN 기반 IDS를 제안한다.

그리고 비지도학습의 경우 Auto Encoder (AE)는 두 개의 AE를 사용하여 초기 비지도학습에서 정상 흐름과 공격 흐름을 별도로 훈련하고, 이러한 재구성된 샘플을 지도학습 단계에서 1차원 컨볼루션 신경망 (1D-CNN)에 적용한다 [27]. 또한 Stacked AE [28, 29], Sparse AE [29, 30] 및 AE의 변형 모델 [31, 32]과 같은 다양한 AE의 변형도 관련된 많은 다른 연구에서 IDS를 구현하기 위한 방법으로 사용된다.

본 논문에서 제안하는 방법이 관련 연구와 갖는 주요한 차이점은 다음과 같다. 첫째, [24]를 중심으로 기존 연구보다 높은 분류 성능 (F1 score)으로 비정상적인 트래픽을 탐지한다. 둘째, 본 논문은 비정상 트래픽의 유형을 지정하기 위해 다중 클래스 분류를 제공한다. 셋째, 대부분의 연구가 성능 향상을 위해 NSL-KDD와 같은 특정 1개의 공개 데이터셋을 대상으로 하는 반면, 본 연구에서는 특정 데이터셋을 대상으로 하지 않고 서로 다른 feature로 구성된 3가지 다른 공개 데이터셋에 대해 모두 고르고 우수한 성능을 보여준다. 마지막으로, 다양한 baseline 기계학습 알고리즘을 체계적으로 탐색하여 제안하는 모델의 성능을 비교한다.

Ⅲ. 딥러닝 기반 네트워크 공격 및 침입 탐지 모델

3.1. 네트워크 공격 및 침입 탐지 모델 구조

본 문서에서 제안하는 딥러닝 기반 네트워크 공격 및 침입 탐지 모델은 [33]의 VM 및 서비스의 이상을 탐지하는 선행 연구 모델을 기반으로 하며, 그림 1과 같이 설계되어 있다. 해당 모델은 네트워크 트래픽 통계 정보를 입력으로 받아 네트워크 공격이나 침입 등 비정상 트래픽을 정상 트래픽과 비교하여 분류한다. 해당 탐지 (분류) 문제는 출력

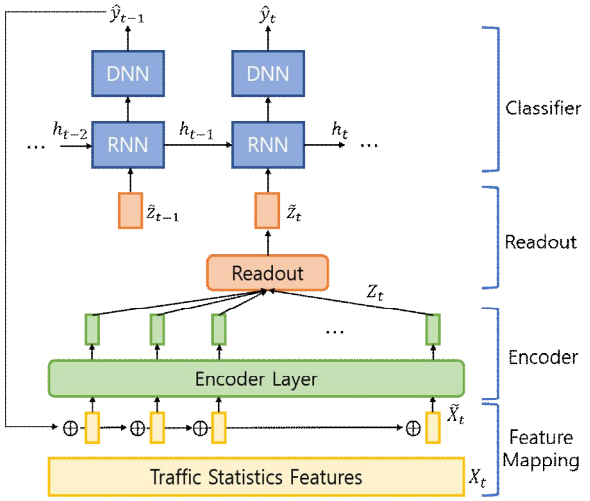


그림 1. 딥러닝 기반 네트워크 공격 및 침입 탐지 모델 구조도
Fig. 1. Design of Deep Learning-based Network Attack&Intrusion Detection Model

$\hat{y}_t$를 생성하는 매개변수 함수 $f(X_t;\theta)$로 볼 수 있다 (X는 입력값, t는 시간 인덱스, θ는 매개변수를 의미한다). 이때, 이진 분류의 경우 출력은 정상 트래픽과 비정상 트래픽을 나타내는 0과 1로 구분되며, 다중 클래스 분류의 경우에는 클래스별 숫자 값으로 구분된다.

네트워크 공격 및 침입 탐지 모델이 사용하는 딥러닝 알고리즘은 네트워크 트래픽의 시계열 특성을 보다 잘 학습할 수 있도록 현재 시간 인덱스의 입력과 이전 시간 인덱스의 입력값을 모두 고려한다. 따라서 현재 시간 인덱스 t에 대한 입력은 $\overset{t}{\underset{t-l+1}{X}} = [X_{t-l+1}, X_{t-l+2}, \cdots, X_t]$이며, 여기서 l은 윈도우 사이즈 (학습 시 지정한 입력 시퀀스 길이)를 의미한다. 제안된 모델은 이와 같은 입력 시퀀스를 기반으로 학습을 진행하며, 학습 및 분류는 그림 1과 같이 총 4개의 계층을 통해 진행된다.

먼저, Feature Mapping 계층은 활성화 함수 (activation function)가 없는 fully connected layer로 구성되어 입력 벡터 X_t를 $\tilde{X}_t \in R^D$로 매핑한다. 여기서 D는 입력값의 차원 (dimension)을 의미한다. 그리고 입력값으로 활용되는 데이터는 네트워크 트래픽 플로우의 통계 정보를 포함하는 각 feature 값들로 구성된다. 따라서 이를 수식화하면 아래 수식과 같이 정의된다.

$$\tilde{X}_t = feature\_mapping(X_t;\theta_{mapping})$$

다음으로, Encoder layer는 feature mapping layer에서 처리되어 Encoder layer로 입력된 전체 데이터를 축소시킨 벡터로 변환한다. 이 과정에서 활용되는 알고리즘은 RNN 계열의 알고리즘인 기본 RNN과 Bi-directional RNN (Bi-RNN) [6]이나 Encoder/Decoder 구조를 지니는 Transformer [7]이다. 이를 통해 전체 데이터를 축소하여 입력 벡터의 크기를 줄임으로써 학습이 용이하게 이루어질 수 있도록 한다. Encoder layer에서 활용되는 알고리즘은 학습 시 Encoder layer의 매개변수 지정을 통해 3개 알고리즘 중 1개를 선택하여 학습을 진행할 수 있다. Encoder layer의 학습 과정을 수식화하면 아래 수식과 같이 정의된다. 여기서 Z_t는 Encoder layer를 거쳐 출력된 데이터를 나타낸다.

$$Z_t = Encoder(\tilde{X}_t;\theta_{encoder})$$

그리고 Readout layer는 데이터를 하나의 벡터로 축약한다. 해당 계층에서 사용되는 readout 함수는 min/max/self-attention 3개 함수가 존재한다. 해당 함수들은 Encoder layer에서와 마찬가지로 매개변수 설정을 통해 3개 함수 중 1개 함수를 사용자가 선택할 수 있다. Encoder layer에서 RNN이나 Bi-RNN 알고리즘을 사용하는 경우 max/min 함수와 같이 일반적으로 많이 사용되는 pooling 방법을 사용하고, Transformer를 사용하는 경우 self-attention을 통해 가중치 합 (weighted sum)을 활용하여 Encoder layer에서 출력된 벡터 집합을 축약한다 [34]. Readout layer의 출력은 $\tilde{Z}_t$로 표현되며, 수식은 아래와 같이 정의된다.

$$\tilde{z}_t = Readout(Z_t;\theta_{readout})$$

마지막 classifier layer는 RNN과 DNN이 결합된 형태로 구성된다. 해당 layer는 시간 단계에 따라 먼저 RNN 모델은 입력 시퀀스 $[\tilde{z}_{t-l+1}, \tilde{z}_{t-l+2}, \cdots, \tilde{z}_t]$을 받아 hidden state $[h_{t-l+1}, h_{t-l+2}, \cdots, h_t]$를 출력한다. 그 후, DNN은 RNN의 hidden state h_t를 입력으로 받아 탐지 결과 $\tilde{y}_t = DNN(h_t)$를 출력한다. 해당 과정의 순차적인 식은 다음과 같이 정의된다:

$$\overset{t}{h}_{t-l+1} = RNN(\overset{t}{\tilde{z}}_{t-l+1};\theta_{RNN}),$$

$$\hat{y_t} = DNN(h_t;\theta_{DNN})$$

그리고 classifier layer에서 분류를 위한 목적함수 J는 다음과 같이 주어진다:

$$J(y_t, \hat{y_t}) = -y_t\log(\hat{y_t}) - (1-y_t)\log(1-\hat{y_t})$$

또한, 제안하는 모델의 성능 향상을 위해 이전 시간 단계의 예측 결과를 사용한다. 네트워크 공격 및 침입과 같은 비정상적인 이벤트는 이전 시간 인덱스에서 발생한 이벤트와 높은 상관관계가 있다 [4], [33]. 따라서 제안하는 모델은 선행 시간 인덱스에서의 예측 결과를 현재 시간 인덱스의 트래픽 분류에 활용하며, 현재 입력 X_t와 함께 이전 예측 결과 $\hat{y}_{t-1}$를 사용하도록 모델의 입력을 다음과 같이 수정한다:

$$X_t = \hat{y}_{t-1} \oplus X_t$$

3.2. 네트워크 공격 및 침입 탐지 시스템 설계

상기 설명한 네트워크 공격 및 침입 탐지 모델을 실제 네트워크 환경에서 구현 및 적용하기 위한 네트워크 공격 및 침입 탐지 시스템의 설계는 그림 2와 같다. 제안된 설계는 본 연구 과제를 기반으로 NFV 환경에서 운용이 가능한 물리 네트워크와 가상 네트워크를 타겟으로 한다.

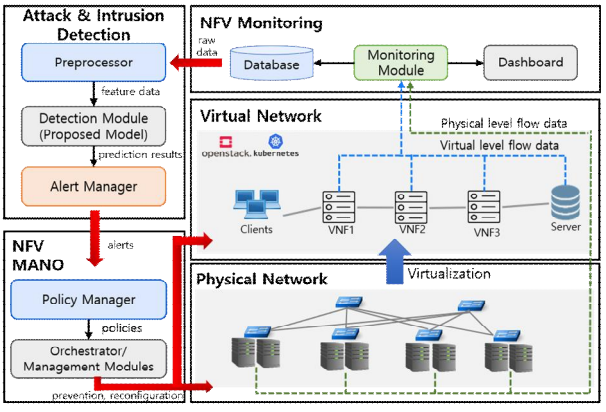


그림 2. NFV 환경을 위한 네트워크 공격 및 침입 탐지 시스템 구조도
Fig. 2. Design of Network Attack&Intrusion Detection System for NFV environments

NFV 환경은 서버 및 스위치를 활용하여 물리 네트워크를 구성하고, 이후 OpenStack/Kubernetes와 같은 VIM (Virtual Infrastructure Manager)를 이용하여 가상 네트워크를 구축한다. 가상 네트워크에서 VNF/CNF 인스턴스들을 동작시켜 서비스를 운용한다. NFV 환경에서 모니터링 기능은 물리 및 가상 서버의 CPU, 메모리와 같은 자원 사용 정보와 네트워크 트래픽 정보, syslog와 같은 로그 정보를 포함한다. 공격 및 침입 탐지 모델에 사용되는 네트워크 트래픽 정보는 플로우 단위로 수집한 통계 정보를 포함하며, 수집된 데이터는 모니터링 모듈로 전달되어 데이터를 대시보드를 통해 실시간 모니터링 정보를 시각화하여 제공하거나, 데이터베이스에 시계열 형태의 데이터로 저장된다.

물리 및 가상 네트워크에 대한 모니터링 모듈을 통해 수집된 트래픽 통계 데이터는 데이터베이스에 저장된 데이터 형태로 공격 및 침입 탐지 모듈에 전달된다. 이 raw 데이터는 전처리기를 통해 모듈화된 탐지 모델에서 사용할 수 있는 형식으로 변환된다. 다음으로, 공격 및 침입 탐지 모듈은 전처리된 데이터를 입력으로 사용하고 탐지 결과를 출력한다. 트래픽 데이터는 정상 또는 네트워크 공격/침입으로 분류하고, 그 결과를 기반으로 Alert Manager는 정책 관리자에게 네트워크 공격 또는 침입 트래픽이 발생했음을 알린다.

이후 전체 네트워크 환경에 대한 관리 정책을 결정하는 Policy Manager는 공격 대상 주소에 대한 보안 규칙을 업데이트하거나 탐지 결과에 따라 서비스 품질 또는 보안 수준을 유지하기 위한 새로운 관리 정책을 생성한다. 마지막으로 정책 관리자는 관리 정책을 Orchestrator로 전송하여 네트워크에 정책을 반영한다. 필요한 경우 Orchestrator는 Auto-Scaling 또는 Live Migration과 같은 다른 관리 모듈과 함께 네트워크를 관리한다. 제안된 방법을 통해 네트워크 공격 및 탐지를 위한 통합적인 폐쇄 루프 자동화 (closed-loop automation)로 구성된 시스템을 기대할 수 있다.

Ⅳ. 결과 및 검증

4.1. 실험 환경

본 연구에서 제안하는 공격 및 침입 탐지 모델의 성능을 평가하기 위해, 네트워크 공격 및 침입과 관련된 공개 데이터셋을 사용하여 성능을 검증한다.

Datasets		NSL-KDD [5]	UNSW-NB15 [6]	CICIDS 2017 [7]
No. Features		41	47	78
No. Records	Total	Normal (77,054) Abnormal (71,463)	Normal (93,000) Abnormal (164,673)	Normal (2,273,098) Abnormal (557,646)
	Train	Normal (67,343) Abnormal (58,630) - DoS (45,927) - Probe (11,656) - R2L (995) - U2R (52)	Normal (37,000) Abnormal (45,332) - Worms (44) - Analysis (677) - Backdoor (583) - DoS (4,089) - Exploits (11,132) - Fuzzers (6,062) - Generic (18,871) - Reconnaissance (3,496) - Shell Code (378)	Normal (2,273,098) Abnormal (557,646) - FTP-Patator (7,938) - SSH-Patator (5,897) - DoS GoldenEye (10,293) - DoS Hulk (231,073) - DoS Slowhttptest (5,499) - DoS slowloris (5,796) - Heartbleed (11) - XSS (652) - SQL (21) - Brute (1,507) - Infiltration (36) - DDoS (128,027) - PortScan (158,930) - Botnet (1,966)
	Test	Normal (9,711) Abnormal (12,833) - DoS (7,458) - Probe (2,421) - R2L (2,754) - U2R (200)	Normal (56,000) Abnormal (119,341) - Worms (130) - Analysis (2,000) - Backdoor (1,746) - DoS (12,264) - Exploits (33,393) - Fuzzers (18,184) - Generic (40,000) - Reconnaissance (10,491) - Shell Code (1,133)	

그림 3. 성능 검증에 사용된 공개 데이터셋의 세부 정보
Fig. 3. Details of the public datasets

성능 검증을 위한 데이터셋은 NSL-KDD [5], UNSW-NB15 [6] 및 CICIDS 2017 [7]의 3가지 데이터셋이 사용된다. NSL-KDD는 IDS에서 수집된 인터넷 트래픽 데이터를 포함하고 있으며, 4가지 공격 종류로 구성된다. UNSW-NB15는 NSL-KDD에 포함되지 않은 low footprint와 같은 공격 시나리오가 포함되어 있으며, 총 9개의 서로 다른 공격 종류로 구성된다. 그리고 CICIDS 2017는 실제 데이터와 유사한 benign (normal) 및 최신 네트워크 공격이 포함되어 있으며, 총 14개의 공격 종류로 구성되어 있다. 상기 3가지 데이터셋에 대한 공격 종류 및 데이터 수 등의 상세 정보는 그림 3과 같다.

4.2. 실험 시나리오

본 연구의 실험은 모델의 성능을 두 가지 방식으로 평가한다. 먼저 이진 분류를 통해 정상 트래픽 데이터와 비정상 트래픽 데이터를 잘 분류할 수 있는지 평가한다. 그리고 분류 문제에서 최근 일반적으로 사용되는 머신러닝 알고리즘 (SVM, Random Forest, XGBoost, DNN)으로 구성된 baseline 모델과 성능을 비교한다. baseline 모델은 Python, Keras, Scikit-learn을 사용하여 구현된다. 다음으로, 보다 상세한 분류 정보를 제공하기 위해 다중 클래스 분류를 수행하여 성능을 분석한다.

NSL-KDD 및 UNSW-NB15의 경우 모델의 성능을 평가하기 위해 학습에 활용하기 위한 training set과 모델의 성능을 평가하기 위한 test set이 제공된다. 따라서 다른 연구와 정확한 성능 비교를 위해 제공된 데이터셋을 사용한다. 해당 데이터셋들은 학습 시 training set을 training set과 validation set으

Method	Model		NSL-KDD	UNSW-NB15	CICIDS 2017
	Encoder	Readout			
Baseline	SVM		0.6402	0.8391	0.9286
	Random Forest		0.6527	0.8681	0.9858
	XGBoost		0.6977	0.8650	0.9899
	DNN		0.5453	0.8472	0.9198
	R. Vinayakumar et. al [24]		0.8070	0.8200	0.9390
Proposed (w/o previous prediction)	RNN	Max / Mean / Self-attention	0.9253 / 0.9238 / **0.9338**	0.9188 / 0.9167 / **0.9193**	0.9825 / **0.9838** / 0.9812
	Bi-RNN		0.9285 / **0.9364** / 0.9363	0.9206 / 0.9186 / **0.9224**	0.9836 / 0.9841 / **0.9843**
	Transformer		0.9223 / **0.9328** / 0.9267	**0.9215** / 0.9197 / 0.9192	0.9889 / 0.9880 / **0.9890**
Proposed (w/ previous prediction)	RNN	Max / Mean / Self-attention	0.9461/ 0.9403 / **0.9561**	0.9274 / 0.9282 / **0.9287**	**0.9897** / 0.9888 / 0.9894
	Bi-RNN		**0.9552** / 0.9428 / 0.9440	0.9279 / 0.9296 / **0.9377**	0.9913 / 0.9905 / **0.9917**
	Transformer		**0.9563** / 0.9450 / 0.9551	0.9324 / 0.9316 / **0.9346**	0.9952 / **0.9966** / 0.9955

그림 4. 이진 분류 성능 비교 결과 (F1 score)
Fig. 4. Results of performance comparison of binary classification (F1 score)

로 나누어 모델의 학습 과정에서의 성능을 1차적으로 검증한다. 하지만 CICIDS 2017의 경우 training set과 test set를 제공하지 않는다. 또한 해당 데이터셋은 4:1의 비율로 정상 및 비정상 데이터의 불균형이 다른 데이터셋에 비해 심하기 때문에 전처리 과정에서 정상 트래픽 데이터에 대해 under-sampling을 수행하여 data imbalance 문제를 약 2:1 정도로 완화시킨다. 그 후, 데이터셋을 75%의 training set과 25%의 test set으로 나눈다. 이때, training set의 경우 NSL-KDD 및 UNSW-NB15와 같은 방법으로 training set과 validation set으로 나누어 학습을 진행한다.

하지만 모든 데이터셋에서 이진 분류에서도 각 레이블 간의 데이터 불균형이 여전히 존재하고, 다중 클래스 분류에서는 그림 3과 같이 불균형이 크게 존재한다. 따라서 정확도 (accuracy) 지표를 사용할 경우 데이터의 불균형으로 인해 부정확한 성능 평가가 발생할 가능성이 존재하므로 보다 정확한 성능 평가를 위해 precision과 recall 값의 조화평균으로 데이터 불균형을 고려하는 지표인 F1 score를 성능 평가에 사용한다. 실험 결과는 실험을 10회 반복하여 F1-score의 평균을 계산한다.

4.3 실험 결과

1) 이진 분류 (Binary Classification): 그림 4는 3개 공개 데이터셋에 대한 각 모델의 성능을 설명한다. 먼저, NSL-KDD 데이터셋의 경우 baseline 모델은 0.53~0.69로 낮은 정확도 (F1-score)를 보였다. 관련 연구 [24]의 경우 약 0.8070로 가장 높은 성능을 보였다. 반면 본 연구의 모델은 previous prediction 기능을 사용하지 않은 경우, 0.92에서 0.93으로 상대적으로 높은 성능을 보였다. encoder 및 readout layer의 하이퍼파라미터 설정에 따라 F1-score 값이 다소 차이가 있지만, 제안하는 모델은 encoder layer에서 Bi-RNN을 사용하고 readout layer에서 mean 함수를 사용할 때 0.9338의 가장 높은 F1-score를 보였다. 또한, previous prediction 기능을 사용하는 경우 해당 기능을 사용하지 않은 결과에 비해 성능이 향상되었다. 특히, 제안된 모델은 encoder layer에서 Transformer를 사용하고 readout layer에서 max 함수를 사용할 때 0.9563의 가장 높은 성능을 보였다.

다음으로 UNSW-NB15 데이터셋의 경우 NSL-KDD와 달리 baseline 모델의 성능도 최대 0.8681의 성능을 보였다. baseline 모델 중 Random Forest 모델과 XGBoost 모델은 가장 높은 성능을 보였지만 분류 정확도 (F1-score) 0.9를 달성하지 못하였다. 반면 제안하는 모델은 NSL-KDD보다 약간 낮은 성능을 보였으나 baseline 모델의 성능보다 평균적으로 우수하였다 (약 0.91~0.92). previous prediction 기능을 사용하지 않은 경우 encoder 및 readout layer configuration 중 Bi-RNN과 self-attention을 사용한 결과가 0.9224로 가장 높은 성능을 보였으며, 해당 기능을 사용할 경우 기존보다 개선된 성능을 보였다. 특히 Bi-RNN과 self-attention을 적용한 모델이 0.9377로 가장 성능이 우수하였다.

마지막으로 CICIDS 2017 데이터셋은 대부분의 모델이 높은 분류 정확도를 보였다. baseline 알고리즘의 경우 정확도가 약 0.9 이상이었고, 특히 Random Forest, XGBoost와 같은 DT 계열의 알고

Training setting	Multi-class classification (w/ normal data)			Multi-class classification (w/o normal data)		
Label (No. Record in test set)	RNN (Max / Mean / Self-attention)	Bi-RNN (Max / Mean / Self-attention)	Transformer (Max / Mean / Self-attention)	RNN (Max / Mean / Self-attention)	Bi-RNN (Max / Mean / Self-attention)	Transformer (Max / Mean / Self-attention)
DoS (7,456)	0.894 / **0.901** / 0.894	0.893 / 0.882 / **0.902**	0.897 / 0.889 / **0.899**	0.912 / 0.906 / **0.914**	0.920 / 0.912 / **0.937**	0.897 / 0.902 / **0.905**
Probe (2,421)	0.787 / **0.811** / 0.805	**0.834** / 0.816 / **0.834**	**0.796** / 0.772 / 0.763	**0.832** / 0.820 / 0.807	0.760 / **0.834** / 0.822	**0.813** / 0.810 / 0.808
R2L (2,754)	0.233 / 0.182 / **0.290**	0.200 / 0.185 / **0.224**	0.180 / **0.215** / 0.197	**0.737** / 0.722 / 0.724	**0.736** / 0.720 / 0.703	0.692 / **0.703** / 0.696
U2R (200)	**0.172** / 0.153 / 0.105	0.070 / **0.126** / 0.080	0.160 / 0.144 / **0.218**	0.190 / **0.205** / 0.193	0.172 / 0.208 / **0.216**	0.272 / 0.256 / **0.286**
Normal (9,710)	**0.812** / 0.810 / 0.808	0.802 / 0.804 / **0.826**	0.802 / 0.793 / **0.807**	-	-	-

그림 5. NSL-KDD 데이터셋의 다중 클래스 분류 결과 (F1 score)
Fig. 5. Results of multi-class classification for NSL-KDD dataset (F1 score)

Training setting	Multi-class classification (w/ normal data)			Multi-class classification (w/o normal data)		
Label (No. Record In test set)	RNN (Max / Mean / Self-attention)	Bi-RNN (Max / Mean / Self-attention)	Transformer (Max / Mean / Self-attention)	RNN (Max / Mean / Self-attention)	Bi-RNN (Max / Mean / Self-attention)	Transformer (Max / Mean / Self-attention)
DoS (12,264)	**0.434** / 0.389 / 0.411	**0.435** / 0.425 / 0.393	**0.421** / 0.412 / 0.400	0.602 / 0.642 / **0.657**	0.593 / 0.610 / **0.621**	**0.632** / 0.595 / 0.620
Exploits (33,393)	**0.701** / 0.694 / 0.690	0.655 / 0.650 / **0.682**	**0.683** / 0.679 / 0.676	0.732 / 0.728 / **0.740**	**0.738** / 0.712 / 0.703	**0.781** / 0.770 / 0.752
Fuzzers (18,184)	**0.660** / 0.644 / 0.623	0.605 / **0.640** / 0.638	**0.646** / 0.622 / 0.610	0.730 / 0.740 / **0.751**	0.720/ **0.742** / 0.725	0.725 / 0.740 / **0.761**
Generic (40,000)	**0.966** / 0.950 / 0.953	0.933 / **0.954** / **0.954**	0.942 / 0.952 / **0.976**	0.928 / 0.947 / **0.950**	**0.953** / 0.942 / 0.942	**0.942** / 0.935 / 0.933
Reconnaissance (10,491)	**0.596** / 0.576 / 0.553	0.602 / **0.630** / 0.594	0.622 / **0.643** / **0.640**	0.626 / 0.630 / **0.672**	**0.688** / 0.662 / 0.654	0.660 / **0.687** / 0.658
Shell code (1,133)	0.025 / **0.032** / **0.032**	0.014 / 0.020 / **0.052**	0.215 / **0.301** / 0.260	0.120 / 0.132 / **0.155**	0.127 / 0.136 / **0.209**	0.200 / 0.252 / **0.335**
Normal (55,989)	0.930 / 0.923 / **0.937**	0.926 / **0.938** / **0.938**	**0.945** / 0.932 / 0.937	-	-	-

그림 6. UNSW-NB15 데이터셋의 다중 클래스 분류 결과 (F1 score)
Fig. 6. Results of multi-class classification for UNSW-NB15 dataset (F1 score)

리즘은 0.98 이상의 높은 성능을, DNN 모델은 약 0.92의 성능을 보였다. 본 연구의 모델은 previous prediction 기능을 사용하지 않았을 때 0.98 이상의 분류 정확도를 보였고, 이러한 결과는 DT를 기반으로 한 baseline 모델 (Random Forest, XGBoost)과 유사하였다. 또한, previous prediction을 사용했을 경우 약 0.99 이상의 F1-score를 보였다. 특히, encoder layer에서 Transformer를 사용하고 readout layer에서 mean 함수를 사용할 때 0.9966으로 가장 높은 분류 성능을 보였다. 따라서 성능 측정 결과, 제안하는 모델이 previous prediction 기능을 사용할 경우 가장 성능이 우수하였다.

2) 다중 클래스 분류 (Multi-class classfication): 다중 클래스 분류 실험은 정상 트래픽 데이터의 포함 유무에 따른 분류 성능을 측정하였다. 먼저 그림 5는 4개의 공격 클래스를 가지는 NSL-KDD 데이터셋에 대한 previous prediction 기능과 함께 제안된 모델로서 공격 유형에 따른 다중 클래스 분류 결과를 보여준다. 정상 트래픽 데이터와 비정상 트래픽 데이터를 함께 분류했을 때 정상 트래픽에 대한 분류 정확도는 약 0.8로 나타났다. 비정상 트래픽 데이터의 경우 DoS는 약 0.9, Probe는 약 0.82를 보인 반면, 데이터 수가 적은 클래스인 R2L (Root to Local)과 U2R (User to Root) 공격은 약 0.2의 낮은 분류 정확도를 보였다.

또한 정상 트래픽 데이터와 비정상 트래픽 데이터에 대해 이진 분류를 먼저 수행하는 시나리오를 가정하고 비정상 트래픽 데이터에 대해 다중 클래스 분류를 수행하였다. 그 결과 DoS 및 Probe 공격의 평균 성능이 약간 향상되었다. 그리고 R2L 공격의 분류 정확도는 약 0.2에서 0.7로 크게 향상되었으며, 전반적으로 정상 트래픽 데이터를 포함하지 않았을 때 더 우수한 성능을 보였다.

다음으로, 그림 6은 UNSW-NB15 데이터셋에 대한 다중 클래스 분류 결과를 나타낸다. 정상 트래픽 데이터를 포함하여 분류를 수행한 경우, 정상 클래스의 분류 정확도는 약 0.93이었다. 여러 공격 클래스 중 Generic 클래스가 0.976으로 가장 높은 분류 정확도를 보였고, Exploits, Fuzzers, Reconnaissance 클래스가 약 0.7 정도의 정확도를 보였다. 하지만 제안하는 모델은 학습 데이터가 매우 적은 3가지 공격 클래스 (training dataset에서 Worms: 44개, Analysis: 677개, Backdoor: 583개 데이터 인스턴스)를 잘 분류하지 못했다.

그리고 정상 트래픽 데이터를 포함하지 않았을 때, NSL-KDD 데이터셋의 결과와 유사하게 평균적

Training setting	Multi-class classification (w/ normal data)			Multi-class classification (w/o normal data)		
Label (No. Record In test set)	RNN (Max / Mean / Self-attention)	Bi-RNN (Max / Mean / Self-attention)	Transformer (Max / Mean / Self-attention)	RNN (Max / Mean / Self-attention)	Bi-RNN (Max / Mean / Self-attention)	Transformer (Max / Mean / Self-attention)
FTP-Patator (1,966)	0.968 / 0.964 / **0.988**	**0.986** / 0.978 / 0.982	**0.988** / 0.986 / **0.988**	0.991 / **0.996** / **0.996**	0.996 / 0.992 / **1.0**	0.998 / **1.0** / 0.998
SSH-Patator (1,451)	0.954 / 0.958 / **0.960**	0.960 / 0.962 / **0.964**	**0.974** / 0.970 / 0.972	0.982 / 0.984 / **0.992**	0.986 / **0.992** / 0.990	0.992 / **0.996** / **0.996**
DoS GoldenEye (2,569)	0.982 / **0.990** / 0.985	**0.990** / 0.988 / **0.990**	0.990 / 0.988 / **0.991**	0.984 / **0.986** / 0.984	0.988 / 0.988 / **0.990**	0.992 / **0.996** / 0.988
DoS Hulk (57,727)	**0.998** / **0.998** / 0.996	**1.0** / **1.0** / 0.998	**1.0** / 0.998 / **1.0**	**1.0** / **1.0** / **1.0**	**1.0** / **1.0** / **1.0**	**1.0** / **1.0** / **1.0**
DoS Slowhttptest (1,347)	0.976 / **0.978** / 0.976	0.982 / 0.976 / **0.986**	0.986 / 0.978 / **0.987**	0.940 / 0.966 / **0.977**	0.934 / 0.972 / **0.982**	**0.980** / 0.972 / 0.962
DoS Slowloris (1,403)	0.948 / 0.951 / **0.954**	0.960 / **0.966** / 0.964	**0.976** / 0.974 / 0.974	0.938 / 0.958 / **0.964**	0.915 / 0.966 / **0.978**	**0.976** / 0.972 / 0.963
XSS (167)	**0.158** / 0.082 / 0.102	**0.118** / 0.048 / 0.002	0 / **0.096** / 0.038	0.200 / 0.236 / **0.272**	0.318 / 0.200 / 0.196	0.69 / 0.384 / **0.385**
Brute force (349)	0.458 / **0.518** / 0.480	0.568 / 0.570 / **0.604**	0.69 / 0.676 / **0.703**	0.740 / **0.758** / 0.718	0.742 / **0.840** / 0.805	**0.896** / 0.854 / 0.879
DDoS (32,008)	0.998 / **1.0** / 0.998	**1.0** / **1.0** / **1.0**	**1.0** / **1.0** / **1.0**	**1.0** / **1.0** / **1.0**	**1.0** / **1.0** / **1.0**	**1.0** / **1.0** / 0.996
PortScan (39,918)	0.972 / 0.966 / **0.978**	0.976 / 0.972 / **0.979**	**0.984** / 0.978 / 0.983	**1.0** / **1.0** / **1.0**	**1.0** / **1.0** / **1.0**	**1.0** / **1.0** / 0.998
Botnet (480)	**0.636** / 0.602 / 0.578	0.624 / 0.586 / **0.648**	0.730 / 0.720 / **0.738**	0.908 / **0.926** / 0.918	**0.948** / 0.922 / **0.948**	0.928 / **0.994** / 0.935
Normal (248,816)	0.990 / 0.990 / **0.991**	0.990 / **0.992** / **0.992**	**1.0** / 0.992 / 0.994	-	-	-

그림 7. UNSW-NB15 데이터셋의 다중 클래스 분류 결과 (F1 score)
Fig. 7. Results of multi-class classification for CICIDS 2017 dataset (F1 score)

으로 성능이 향상되었다. 데이터셋에 존재하는 여러 공격 클래스 중 Generic의 경우 평균 정확도가 비슷했지만, DoS, Exploits, Fuzzers, Reconnaissance, Shell code의 경우 평균 성능 (F1 score)이 약 0.1에서 0.2 정도 향상된 것을 확인하였다. 하지만 Shell code는 약 0.3의 낮은 성능을 보였다. 이러한 결과는 학습 데이터가 매우 적을 때의 NSL-KDD 데이터셋 실험 결과와 유사하였다.

마지막으로, CICIDS 2017 데이터셋의 다중 클래스 분류 결과는 그림 7과 같다. CICIDS 2017의 경우 이진 분류에서 0.99의 성능을 보였기 때문에, 여러 클래스 (FTP-Patator, SSH-Patator, DoS 관련 공격 클래스)에서는 탐지 정확도가 F1 Score 기준 1에 가까운 결과를 보였다. XSS (Cross-Site Scripting) 공격은 약 0.1의 낮은 성능을 보였지만 정상 트래픽 데이터를 포함하지 않았을 때에는 그 성능이 약 0.38까지 향상되었다. 그리고 brute force 공격과 botnet 공격은 평균 약 0.65 정도의 분류 정확도를 보였으나, 해당 클래스들의 분류 성능도 정상 데이터를 제외했을 때 약 0.2 이상 개선되었다 (Brute force: 0.896, Botnet: 0.994).

4.4 결과 분석

먼저 이진 분류 결과에서, 본 연구의 모델은 3가지 서로 다른 feature를 지니는 데이터셋에서 baseline 모델보다 높은 성능을 보였다. 특히 previous prediction 기능을 사용했을 때 가장 높은 성능 정확도를 보였기 때문에 시계열 특성을 지니는 트래픽 데이터의 학습에 강점을 지니는 것을 확인하였다. 그리고 NSL-KDD 및 UNSW-NB15의 경우 baseline model과 유의미한 성능 차이를 보였고, CICIDS 2017 데이터셋의 경우 baseline 모델보다 비슷하거나 높은 성능을 보였다. 따라서 baseline 모델들과 [24]와 같은 관련 연구는 데이터셋마다 성능의 차이가 존재하지만 본 연구의 모델은 3가지 서로 다른 데이터셋에서도 고르게 우수한 성능을 보이는 것을 확인하였다.

다음으로, 다중 클래스 분류에서 모델은 학습 데이터가 충분할 때 클래스를 잘 분류하였다. 그러나 NSL-KDD에서 1,000개 미만 (R2L: 995개 데이터) 또는 100개 미만 (U2R: 52개 데이터)과 같이 학습 데이터가 매우 작은 경우 학습이 잘 수행되지 않고 분류 정확도가 낮았다. 이러한 문제는 NSL-KDD뿐만 아니라 UNSW-NB15 (Worms, Analysis, Backdoors, Shell code) 및 CICIDS 2017 (Heartbleed, XSS, SQL injection, infiltration)에서도 확인할 수 있다. 이 문제는 관련 연구에서도 데이터 개수 부족으로 인한 학습 문제가 발생하였는데, 대개 소수의 데이터로 구성된 클래스의 데이터를 그대로 학습시킨 경우에 발생하였다. 해당 문제는 제안하는 모델의 한계점이기도 하므로 개선을 위한 향후 연구에서는 SMOTE (Synthetic Minority Over-sampling Technique)와 같은 over-sampling 기법을 특정 클래스에 적용하여 다중 클래스 분류 성능을 개선할 수 있다. 또한, over-sampling 기법은 over-fitting의 위험성이 존재하기 때문에 모델 학습 시 각 클래스에 대한 학습 가중치 (weight) 매개변수를 제어하는 것도 또 다른 솔루션이 될 수 있다.

추가적으로 비정상 트래픽을 보다 자세하게 분석하기 위한 Root-Cause Analysis (다중 클래스 분류)에서, 먼저 트래픽을 정상과 비정상으로 이진 분류한 후 비정상 트래픽의 유형을 세분화하여 분류하는 것이 더 높은 정확도를 지니는 것을 확인하였다. 이는 Confusion Matrix를 통한 결과 분석에서 대부분 잘못 분류된 데이터가 정상 트래픽으로 분류되어 발생하는 것으로 나타났다. 따라서 향후 연구를 위해 이중 구조를 지니는 탐지 모델을 사용하는 방법도 고려될 수 있다.

Ⅴ. 결 론

네트워크 및 서비스가 발전하면서 다양한 공격 및 침입에 대한 위협이 증가하였다. 본 논문에서는 네트워크 공격 및 침입과 관련된 비정상 트래픽을 탐지하기 위해 딥러닝을 기반으로 하는 탐지 모델을 제안한다. 제안하는 모델은 3가지 서로 다른 공개 데이터셋에서 정상 데이터와 비정상 데이터의 이진 분류 및 Root-Cause Analysis를 위한 다중 클래스 분류를 수행한다. 실험 결과는 이진 분류에서의 F1-score를 기준으로 볼 때, NSL-KDD에서는 0.956, UNSW-NB15에서는 0.938, 그리고 CICIDS 2017에서는 0.997의 결과값을 가지며, 이는 baseline 모델 및 관련 연구와 비교했을 때 보다 향상되고 균일한 성능을 보인다.

향후 연구에서는 제안하는 모델의 성능을 향상시키고, 네트워크 공격 및 침입과 관련된 보다 다양한 데이터셋을 이용하여 성능을 검증할 예정이다. 또한, NFV 기반의 실제 테스트베드 환경 [35]으로의 적용을 통해 실제 환경에서 생성된 정상 및 비정상 트래픽 데이터를 사용하여 모델의 성능을 검증할 예정이다.

References

[1] R. Boutaba et al., ″A comprehensive survey on machine learning for networking: evolution applications and research opportunities″, in Journal of Internet Services and Applications, vol. 9, issue 1, 2018.

[2] Kwon, D. et al., ″A survey of deep learning-based network anomaly detection″, Cluster Computing vol.22, pp. 949–961, Sep. 2019.

[3] N. Moustafa, J. Hu, and J. Slay, ″A holistic review of Network Anomaly Detection Systems: A comprehensive survey,″ Journal of Network and Computer Applications, Vol. 128, pp. 33-55, 2019.

[4] Y. Bengio, R. Ducharme, and P. Vincent, ″A Neural Probabilistic Language Model″, Advances in neural information processing systems, vol. 13, 2000.

[5] ISCX NSL-KDD dataset 2009, ″Nsl-kdd data set for network-based intrusion detection systems,″ [Online]. Available at http://nsl.cs.unb.ca/KDD/NSLKDD.html, Mar. 2009.

[6] N. Moustafa and J. Slay, ″UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set),″ 2015 Military Communications and Information Systems Conference (MilCIS), pp. 1-6, 2015.

[7] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, ″Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization″, 4th International Conference on Information Systems Security and Privacy (ICISSP), pp. 108-116, Portugal, Jan. 2018.

[8] Anwar, Shahid, et al. ″From intrusion detection to an intrusion response system: fundamentals, requirements, and future directions.″ Algorithms 10.2 (2017): 39.

[9] Ahmad, Tanwir, et al. ″Early Detection of Network Attacks Using Deep Learning.″ arXiv preprint arXiv:2201.11628 (2022).

[10] Javaid, Ahmad, et al. ″A deep learning approach for network intrusion detection system.″ Eai Endorsed Transactions on Security and Safety 3.9 (2016): e2.

[11] Chalapathy, Raghavendra, and S. Chawla, ″Deep learning for anomaly detection: A survey.″ arXiv preprint arXiv:1901.03407, 2019.

[12] Schuster, Mike, and Kuldip K. Paliwal. ″Bidirectional recurrent neural networks.″ IEEE transactions on Signal Processing 45.11 (1997): 2673-2681.

[13] Vaswani, Ashish, et al. ″Attention is all you

need." Advances in neural information processing systems 30 (2017).

[14] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.

[15] Ahmad, Zeeshan, et al. "Network intrusion detection system: A systematic study of machine learning and deep learning approaches", Transactions on Emerging Telecommunications Technologies vol. 32(1), 2021.

[16] Chary S, Rama B., "A survey on comparative analysis of decision tree algorithms in data mining", International Journal of Advanced Scientific Technologies, Engineering and Management Sciences, vol. 3, pp. 91-95, 2017.

[17] R. Primartha and B. A. Tama, "Anomaly detection using random forest: A performance revisited," 2017 International Conference on Data and Software Engineering (ICoDSE), pp. 1-6, 2017.

[18] F. H. Kusumaputri and A. S. Arifin, "Anomaly Detection based on NSL-KDD using XGBoost with Optuna Tuning," 2022 7th International Conference on Business and Industrial Research (ICBIR), pp. 586-591, 2022.

[19] Ghanem, Kinan, et al. "Support vector machine for network intrusion and cyber-attack detection." 2017 sensor signal processing for defence conference (SSPD), 2017.

[20] Ma Z, Kaban A. K-Nearest-Neighbours with a novel similarity measure for intrusion detection. Paper presented at, Proceedings of the IEEE 13th UK Workshop on Computational Intelligence (UKCI), Guildford, UK, pp.266-271, 2013.

[21] Zhang Y, Cao G, Wang B, and Li X., "A novel ensemble method for k-nearest neighbor", Pattern Recogn., vol. 85, pp. 13-25, 2019.

[22] Li, Zhengjie, Yongzhong Li, and Lei Xu. "Anomaly intrusion detection method based on k-means clustering algorithm with particle swarm optimization." 2011 international conference of information technology, computer engineering and management sciences, vol. 2, 2011.

[23] R. S. Naoum, N. A. Abid, and Z. N. Al-Sultani,"An Enhanced Resilient Backpropagation Artificial Neural Network for Intrusion Detection System," International Journal of Computer Science and Network Security, vol. 12, no. 3, pp. 11–16, 2012.

[24] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat and S. Venkatraman, "Deep Learning Approach for Intelligent Intrusion Detection System," in IEEE Access, vol. 7, pp. 41525-41550, 2019.

[25] Mittal, Mohit, et al. "Analysis of security and energy efficiency for shortest route discovery in low-energy adaptive clustering hierarchy protocol using Levenberg-Marquardt neural network and gated recurrent unit for intrusion detection system." Transactions on Emerging Telecommunications Technologies, vol. 32, issue. 6, 2021.

[26] C. Xu, J. Shen, X. Du and F. Zhang, "An Intrusion Detection System Using a Deep Neural Network With Gated Recurrent Units," in IEEE Access, vol. 6, pp. 48697-48707, 2018.

[27] G. Andresini, A. Appice, N. D. Mauro, C. Loglisci and D. Malerba, "Multi-Channel Deep Feature Learning for Intrusion Detection," in IEEE Access, vol. 8, pp. 53346-53359, 2020.

[28] F. A. Khan, A. Gumaei, A. Derhab and A. Hussain, "A Novel Two-Stage Deep Learning Model for Efficient Network Intrusion Detection," in IEEE Access, vol. 7, pp. 30373-30385, 2019.

[29] B. Yan and G. Han, "Effective Feature Extraction via Stacked Sparse Autoencoder to Improve Intrusion Detection System," in IEEE Access, vol. 6, pp. 41238-41248, 2018.

[30] M. Al-Qatf, Y. Lasheng, M. Al-Habib and K. Al-Sabahi, "Deep Learning Approach Combining Sparse Autoencoder With SVM for Network Intrusion Detection," in IEEE Access, vol. 6, pp. 52843-52856, 2018.

[31] Malaiya, Ritesh K., et al. "An empirical evaluation of deep learning for network anomaly detection." 2018 IEEE International

Conference on Computing, Networking and Communications (ICNC), 2018.
[32] Y. Yang, K. Zheng, B. Wu, Y. Yang and X. Wang, "Network Intrusion Detection Based on Supervised Adversarial Variational Auto-Encoder With Regularization," in IEEE Access, vol. 8, pp. 42169-42184, 2020.
[33] C. Lee, J. Hong, D. Heo, H. Choi, "Sequential Deep Learning Architectures for Anomaly Detection in Virtual Network Function Chains", The 12th International Conference on ICT Convergence (ICTC 2021), Jeju, Oct. 20-22, 2021.
[34] H. Choi, K. Cho, and Y. Bengio, "Fine-grained attention mechanism for neural machine translation," Neurocomputing, vol. 284, pp. 171 –176, 2018.
[35] DPNM, "Network Intelligence Project," [Online]. Available at: https://github.com/dpnm-ni.

홍 지 범 (Jibum Hong)

2017 한양대학교 ERICA 컴퓨터공학과 학사
2017 ~ 2020 포항공과대학교 컴퓨터공학과 석사
2020 ~ 현재 포항공과대학교 컴퓨터공학과 박사과정
<관심분야> SDN/NFV, 네트워크 관리, 머신러닝

최 민 지 (Minji Choi)

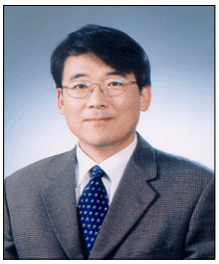
2021 한양대학교 ERICA, 전자공학부 학사
2021 ~ 현재 포항공과대학교 인공지능대학원 석사과정
<관심분야> SDN/NFV, 네트워크 관리, 머신러닝

유 재 형 (Jae-Hyoung Yoo)

1983 연세대학교 전자공학과 학사
1985 연세대학교 전자공학과 석사
1999 연세대학교 컴퓨터공학과 박사

1986 ~ 2012 KT 네트워크 연구소
2012 ~ 2013 KAIST 전기 및 전자공학과 연구교수
2013 ~ 2016 포항공과대학교 컴퓨터공학과 연구교수
2016 ~ 2018 정보통신기술진흥센터 네트워크 CP
2018 ~ 현재 포항공과대학교 컴퓨터공학과 연구교수
<관심분야> 네트워크 관리 및 보안, SDN, OpenFlow

홍 원 기(James Won-Ki Hong)



2020~현재 포스텍 크립토블록체인연구센터 공동센터장
2019-현재 포스텍 교육혁신센터장
2012~2014 KT 종합기술원 원장
2008~2012 포스텍 정보전자융합공학부장
2008~2010 포스텍 컴퓨터공학과 주임교수
2007~2010 포스텍 정보통신연구소 연구소장
2007~2011 포스텍 정보통신대학원장
1995~현재 포스텍 컴퓨터공학과 교수
1992~1995 Univ. of Western Ontario, 연구교수
<관심분야> AI 기반 네트워크관리, SDN/NFV, 무크기반 교육, 블록체인 및 암호화폐, Vmeeting 화상회의시스템

https://doi.org/10.22670/knom.2022.25.2.22

분할 NFT 기술에 대한 연구와 활용 방안 분석

최 원 석◆, 우 종 수*, 홍 원 기°

A Study of Fractional NFT and its Applications

Wonseok Choi◆, Jongsoo Woo*, James Won-Ki Hong°

요 약

대체 불가능 토큰 (NFT)은 그 고유성으로 인해 수집품에서부터 시작하여 소유권을 증명하는 용도로 사용되어 왔고 디지털 자산을 넘어 실물 자산에도 활용되고 있다. 블록체인 및 암호화폐와 관련한 여러 사건 사고에도 NFT 시장은 여전히 작지 않으며 그 중 대부분은 이더리움 블록체인에서 발행되고 있다. 하지만 일부 NFT들의 비싼 가격이 사용자들로 하여금 NFT 시장에 쉽게 뛰어들지 못하게 하는 장벽이 되어 왔다. 분할 NFT는 이를 해결하기 위한 것으로 NFT를 여러 개의 조각으로 나누어 각 조각의 가격을 낮추고 유동성을 높인다. 이로 인해 조각 투자 및 분할 소유권 증명이라는 형태로 NFT가 활용될 수 있다. 본 논문에서는 분할 NFT 기술에 대해 연구하고 발행 방법과 그 활용 방안에 대하여 분석한다. 또한, 실제로 분할 NFT가 발행되고 거래되는 플랫폼들에 대하여 소개한다.

Key Words : Blockchain, Smart Contract, Fungible Token, Non-Fungible Token

ABSTRACT

Due to its uniqueness, NFTs have been used to prove ownership, starting with collections, and have been used for real assets beyond digital assets. Despite various incidents related to blockchain and cryptocurrency, the NFT market is still large, and most of them are issued with Ethereum Blockchain. However, the high prices of some NFTs have been a barrier that prevents users from easily entering the NFT market. The fractional NFT can solve this problem. It divides a NFT into multiple pieces to lower the price of each piece and increase liquidity. For this reason, NFT can be used in the form of fractional investment and fractional ownership certification. In this paper, we present the fractional NFT technology, and analyze the minting method and its possible applications. In addition, we introduce platforms on which fractional NFTs can be minted and traded.

※이 논문은 ㈜코인원과 과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구센터육성지원사업의 연구결과로 수행되었음 (IITP-2022-2017-0-01633).
◆ First Author : Center for Crypto Blockchain Research, Department of Computer Science Engineering, Pohang University of Science and Technology (POSTECH), Pohang, South Korea, ws4583@postech.ac.kr
° Corresponding Author : Center for Crypto Blockchain Research, Department of Computer Science Engineering, POSTECH, jwkhong@postech.ac.kr
* Center for Crypto Blockchain Research, POSTECH, woojs@postech.ac.kr
논문번호 : KNOM2022-02-03, Received November 30, 2022; Revised December 10, 2022; Accepted December 16, 2022

Ⅰ. 서 론

블록체인의 스마트 컨트랙트를 통해 발행되는 대체 불가능 토큰 (Non-Fungible Token, NFT)[1]은 기존의 대체 가능한 토큰 (Fungible Token, FT) 과 다르게 각각의 토큰이 고유성을 지닌다. 이러한 특성으로 인해 NFT는 초기에 수집품의 용도로 널리 사용되었으며 대표적인 사례로 크립토키티[2]가 존재한다. 이후 NFT 기술을 다른 방안으로 활용하려는 시도들이 나타났고 NFT를 디지털 자산 및 실물 자산의 소유권을 증명하는 용도로 활용 가능함이 검증되었다.

Coinmarketcap[3]에서 제공하는 통계에 따르면 현재 NFT의 시가 총액은 3조원에 달한다. 또한, Cryptoslam[4]에서 제공하는 통계에 따르면 역대 NFT 거래량은 55조원에 달하며 그 중 이더리움[5] 블록체인을 통해 발행된 NFT의 거래량이 40조원에 달한다. 그 외에도 Ronin[6], Solana[7], Flow[8]등의 블록체인에서도 NFT가 발행되어 거래되고 있으나 여전히 이더리움 블록체인이 NFT 시장의 대부분을 차지하고 있다고 할 수 있다. 한편, NFT 수집품의 경우 Ronin을 기반으로 만들어진 Axie Infinity[9]가 가장 많은 거래량을 차지했고 그 뒤를 Bored Ape Yacht Club[10], CryptoPunks[11] 등이 차지했다.

한편, NFT 시장이 활성화됨에 따라 NFT 자체의 가격 역시 상승해왔으며 고가의 NFT의 경우 유동성이 떨어지게 된다. 이러한 문제를 해결하기 위해 NFT를 분할하는 방안이 등장했다. Fractional NFT, F-NFT라고도 불리는 이러한 방식에서는 고가의 NFT를 별도의 스마트 컨트랙트에 저장하고, 해당 스마트 컨트랙트를 통해 다수의 FT를 발행함으로써 NFT를 여러 개의 FT로 분할한다. 이렇게 생성된 FT를 유통할 경우 각 FT의 소유자들은 자신이 소유한 FT의 비율만큼의 소유권을 획득하는 것이 된다. 따라서 분할 NFT를 통해 소유권을 분할하여 가격을 낮춰 상품의 유동성을 증가시키는 효과가 있다.

분할 NFT를 사용할 경우 필요에 따라 소유자들은 NFT에 따른 권리를 일부 행사하게 할 수 있으며 해당 NFT로부터 발행된 모든 FT를 소유함으로써 본래의 NFT의 소유권을 스마트 컨트랙트로부터 자신에게 이전하는 것이 가능하다. 이러한 기능은 디지털 자산뿐만 아니라 실물 자산을 기반으로 발행된 NFT에서도 효과적으로 활용될 수 있다. 이를 기반으로 분할 NFT를 발행 및 거래할 수 있는 플랫폼들이 다수 존재한다.

하지만 분할 NFT는 이를 만드는 과정에 있어 NFT를 다른 스마트 컨트랙트에 저장해야하기 때문에 보안 문제에 더욱 민감하다. NFT를 분할하여 생성된 FT들이 시장에서 유통되는 동안 스마트 컨트랙트에 저장된 NFT가 유출되거나 변경되는 경우 해당 FT들의 가치가 사라질 수 있기 때문이다. 따라서 NFT를 안전하게 스마트 컨트랙트에 보관할 수 있어야 하며 FT를 통해 NFT를 복원하는 과정 역시 잘 구현되어야 한다.

본 논문에서는 분할 NFT이 어떻게 만들어지고 활용되는가에 대해 분석한다. 대다수의 NFT가 발행되는 Ethereum Request for Comment (ERC)[12] 표준에 대해 정리하고 이를 기반으로 분할 NFT를 발행하는 과정을 분석한다. 그리고 이렇게 발행된 분할 NFT들이 어떻게 활용될 수 있는가에 대해서 분석하며 실제로 분할 NFT가 발행 및 거래되고 있는 플랫폼들에 대해 정리한다.

본 논문의 구성은 다음과 같다. 2장에서는 분할 NFT를 발행함에 있어 필수적인 스마트 컨트랙트와 관련된 ERC 표준들에 대하여 서술한다. 3장에서는 이를 기반으로 분할 NFT를 발행하는 과정과 활용 방안에 대해 설명한다. 4장에서는 분할 NFT와 관련된 플랫폼들에 대해 설명한다. 마지막으로 5장에서는 결론 및 향후 연구 방향에 대해서 다루며 논문을 마무리한다.

Ⅱ. 배 경

2.1. 스마트 컨트랙트

스마트 컨트랙트란 사전에 협의된 내용을 미리 프로그래밍하여 자동으로 작동하게 만들어진 시스템이다. 1세대 블록체인이라 불리는 비트코인[13]과 2세대 블록체인이라 불리는 이더리움의 가장 큰 차이는 스마트 컨트랙트의 도입이라 할 수 있다. 이더리움 네트워크에서는 이더리움 가상 머신(Ethereum Virtual Machine, EVM)[14]을 통해 작성된 스마트 컨트랙트 코드를 실행한다. 이더리움에서 스마트 컨트랙트는 자체 개발된 프로그래밍 언어인 솔리디티(Solidity)[15]를 사용한다. 블록체인 상에 등록된 스마트 컨트랙트는 위조 및 변조가 불가능하기 때문에 이를 기반으로 이루어진 계약을 계약 당사자들

이 신뢰할 수 있다. 하지만 취약점이 발견될 경우 블록체인 네트워크상에 등록된 이후 이를 수정하는 것이 어려워 스마트 컨트랙트를 배포하기 전에 충분한 검증이 필요하다.

2.2. ERC 표준

이더리움 블록체인에는 이더리움의 개선을 위한 제안인 Ethereum Improvement Proposals (EIPs) [16]가 존재한다. 사용자들은 커뮤니티를 통해 자유롭게 이더리움 개선 방안을 제안할 수 있고 합의를 통해 제안이 통과될 수 있다. Ethereum Request for Comment (ERC)는 EIP를 거쳐 통과된 표준의 일부로 개발자들을 위한 어플리케이션 단계에서의 규칙을 정의한 것이다. 대표적으로 스마트 컨트랙트를 통해 발행되는 토큰들에 대한 표준인 ERC-20[17], ERC-721[18], ERC-1155[19] 등이 정의되어있다.

1) ERC-20

ERC-20은 초기에 만들어진 토큰에 대한 표준으로 FT에 대해서 다루며 토큰 전송 및 권한 승인 등이 정의되어 있다. 표 1은 ERC-20 문서에 정의된 함수 일부를 소개한다.

표 1. ERC-20 함수들
Table 1. ERC-20 Functions

Function	Description
function totalSupply() public view returns (uint256)	전체 토큰 수량 반환
function balanceOf(address _owner) public view returns (uint256 balance)	_owner가 보유한 토큰 수량 반환
function transferFrom(address _from, address _to, uint256 _value) public returns (bool success)	_from으로부터 _to에게 _value만큼의 토큰 전송
function approve(address _spender, uint256 _value) public returns (bool success)	_spender에게 함수 실행자의 토큰을 _value만큼 인출할 수 있는 권한 부여

2) ERC-721

ERC-721은 NFT를 위한 표준으로 ERC-20에서 관리할 수 없는 고유한 토큰들을 위해 만들어졌다. ERC-721에는 ERC-20에서 정의된 함수뿐만 아니라 NFT의 소유권과 관련된 함수들이 추가되었다. 표 2는 ERC-721 문서에 정의된 함수 일부를 소개한다.

표 2. ERC-721 함수들
Table 2. ERC-721 Functions

Function	Description
function balanceOf(address _owner) external view returns (uint256)	_owner가 보유한 NFT 수량 반환
function ownerOf(uint256 _tokenId) external view returns (address);	_tokenId를 갖는 NFT의 소유주 반환
function safeTransferFrom(address _from, address _to, uint256 _tokenId) external payable;	_from으로부터 _to에게 _tokenId를 갖는 NFT 전송 _to가 스마트 컨트랙트의 주소인지 검증
function transferFrom(address _from, address _to, uint256 _tokenId) external payable;	_from으로부터 _to에게 _tokenId를 갖는 NFT 전송
function approve(address _approved, uint256 _tokenId) external payable;	_approved에 _tokenId를 갖는 NFT 전송 권한 부여
function setApprovalForAll(address _operator, bool _approved) external;	_operator에게 함수 실행자가 보유한 모든 NFT에 대한 권한 설정
function getApproved(uint256 _tokenId) external view returns (address);	_tokenId를 갖는 NFT의 권한을 갖는 모든 주소 반환
function isApprovedForAll(address _owner, address _operator) external view returns (bool);	_owner의 모든 NFT의 권한을 _operator가 보유하고 있는지 확인

3) ERC-1155

ERC-1155는 여러 종류의 토큰을 지원하기 위한 표준으로 FT, NFT 뿐만 아니라 속성이 변할 수 있는 Semi-Fungible Token 역시 지원한다. ERC-1155를 사용할 경우 여러 종류의 토큰을 하나의 스마트 컨트랙트 만으로 발행할 수 있으며 이들을 한 번에 전송할 수도 있어 거래 비용을 크게 감소시킬 수 있다. 표 3은 ERC-1155 문서에 정의된 함수 일부를 소개한다.

Ⅲ. 분할 NFT

3.1. 분할 NFT 발행 과정

분할 NFT를 발행하는 과정은 그림 1과 같이 크게 2가지 과정으로 구분할 수 있다. 첫 번째는 이미 발행된 NFT를 별도의 스마트 컨트랙트에 전송

표 3. ERC-1155 함수들
Table 3. ERC-1155 Functions

Function	Description
function balanceOf(address _owner, uint256 _id) external view returns (uint256);	_owner가 보유한 _id를 갖는 토큰 수량 반환
function safeTransferFrom(address _from, address _to, uint256 _id, uint256 _value, bytes calldata _data) external;	_from으로부터 _to에게 _id에 해당하는 토큰을 _value만큼 전송
function safeBatchTransferFrom(address _from, address _to, uint256[] calldata _ids, uint256[] calldata _values, bytes calldata _data) external;	_from으로부터 _to에게 _ids에 해당하는 토큰들을 각각 _values만큼 전송
function setApprovalForAll(address _operator, bool _approved) external;	_operator에게 함수 실행자가 보유한 모든 NFT에 대한 권한 설정
function isApprovedForAll(address _owner, address _operator) external view returns (bool);	_owner의 모든 NFT의 권한을 _operator가 보유하고 있는지 확인

해 저장하는 것이다. 이 때 하나의 스마트 컨트랙트 내에 복수의 NFT가 저장되는 것도 가능하며 이 경우 해당 NFT들의 묶음에 대한 FT를 발행하는 것이 된다. 두 번째 과정은 해당 스마트 컨트랙트 내에서 여러 개의 FT를 발행하는 것이다. 예를 들어, 스마트 컨트랙트를 통해 1,000개의 FT를 발행한다면 하나의 FT는 NFT의 1/1,000 만큼의 가치를 지니게 된다. 이렇게 발행된 FT들은 거래를 통해 다른 사람들에게 전송될 수 있으며 FT의 소유자들은 자신이 소유한 지분만큼의 소유권을 얻을 수 있게 된다.

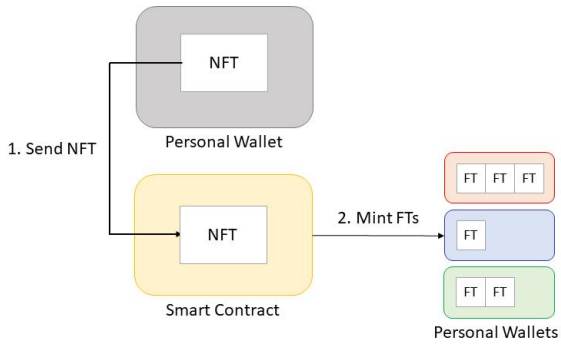


그림 1. 분할 NFT 발행 과정
Fig. 1. Fractional NFT Minting Procedure

3.2. 분할 NFT 구현

현재 대부분의 NFT가 이더리움을 기반으로 발행되는 것을 고려하여 ERC 표준을 기반으로 분할 NFT가 구현된다고 가정 하였을 때, 아래와 같이 4가지 방식의 구현이 가능하다.

1) ERC-721기반의 NFT를 ERC-20 기반의 FT로 분할

2) ERC-721기반의 NFT를 ERC-1155 기반의 FT로 분할

3) ERC-1155기반의 NFT를 ERC-20 기반의 FT로 분할

4) ERC-1155기반의 NFT를 ERC-1155 기반의 FT로 분할

각각의 방식에 있어 구체적인 구현 방식이나 거래 비용이 달라질 수 있기 때문에 ERC 표준에 따른 함수에 대한 충분한 이해가 필요하다. 예를 들어, 앞서 표 2에서 설명한 바와 같이 ERC-721 기반의 스마트 컨트랙트로 발행된 NFT를 safeTransferFrom 함수를 통해 다른 스마트 컨트랙트에 전송할 경우, 해당 함수는 _to에 해당하는 주소가 스마트 컨트랙트에 해당하는 지 확인하게 되는데 이 때 해당 스마트 컨트랙트가 ERC-721 기반의 스마트 컨트랙트에서 발행된 토큰을 소유할 수 있는 지를 확인한다. 이를 통과하기 위해서는 NFT를 저장할 스마트 컨트랙트에 onERC721Received 함수가 정의되어 있어야 한다.

NFT가 스마트 컨트랙트에 저장된 이후에는 특수한 상황을 제외하고 해당 NFT가 이동할 수 없어야 한다. 따라서 스마트 컨트랙트를 설계할 때 NFT가 악의적으로 이동되는 일이 없도록 미연에 방지하거나 이후 FT가 발행되고 거래되는 과정에서 NFT가 제대로 저장되어 있는지 확인하는 과정을 추가하여 안전성을 높일 수 있다.

한편, 추후에 다시 NFT를 얻기 위해서는 특정 조건에 따라 스마트 컨트랙트 내에 저장된 NFT를 사용자에게 전송하는 함수가 구현되어야 한다. 가장 일반적인 방법으로는 발행된 모든 FT를 소유하고 있을 때 이를 확인하여 NFT를 FT의 소유자에게 이동시키는 것이다. 이때, NFT가 올바르게 이전된 이후에는 해당 스마트 컨트랙트에서 발행된 FT들의 가치가 없어지기 때문에 악의적으로 시장에 거래되는 것을 방지하기 위하여 해당 토큰들은 반드시 소각되어야 한다.

3.3. 분할 NFT 활용 방안

분할 NFT를 사용함에 있어 가장 큰 장점은 고가의 NFT를 여러 개의 조각으로 분할함으로써 많은 사람들이 해당 NFT를 나누어 가질 수 있다는 것이다. NFT의 경우 작게는 0.001Ether에서 많게는 수백, 수천의 Ether로도 거래되는 만큼 고가의 NFT는 일반적인 사용자들이 구매하기 어렵다. 하지만 이를 여러 개의 조각으로 분할할 경우 조각 수에 따라 가격이 크게 낮아지게 되고 누구나 조각을 구매할 수 있게 된다. 이러한 점은 상품의 유동성을 높이는 데에도 큰 도움이 된다. 고가의 NFT의 경우 OpenSea[20]와 같은 NFT 거래 플랫폼에서 거래되기 까지 오랜 시간이 걸릴 수 있지만 이를 분할한 토큰의 경우 상대적으로 낮은 가격 및 많은 수량으로 인해 잦은 거래가 발생할 수 있기 때문이다.

분할 NFT는 NFT의 가격이 오를 것으로 예상하여 NFT를 투자의 용도로 구매하고자 할 때 유용하게 사용될 수 있다. 투자를 위한 NFT의 경우 앞서 설명한 바와 같이 고가의 상품일 가능성이 있는데 이러한 NFT가 여러 개의 FT로 분할될 경우 접근성이 높아지기 때문이다. 이는 디지털 자산뿐만 아니라 실물 자산을 기반으로 발행된 NFT에서도 마찬가지로 적용된다.

또한, 가격 상승뿐만 아니라 NFT를 소유함으로써 추가적인 이익이 발생하는 경우에도 유용하다. 대표적으로 집이나 토지를 소유할 경우 이를 임대해주어 추가적인 이익이 발생할 수 있는데 이때 집이나 토지의 소유권을 NFT로 발행하여 임대 수익을 전달할 수 있다. 이러한 NFT들 역시 가격이 높기 때문에 여러 개의 FT로 분할하여 접근성과 유동성을 높이는 것이 가능하다. 결론적으로 분할 NFT는 현재 NFT가 투자 목적의 희귀품 및 소유권으로 사용되는 모든 시장에 활용가능하며 기존 시장보다 더 높은 유동성을 가져다주어 효과적이라 할 수 있다.

Ⅳ. 분할 NFT 플랫폼

본 장에서는 분할 NFT를 발행하거나 거래할 수 있는 플랫폼들에 대해 소개한다.

4.1. Fractional.art

Fractional.art[21]는 현재 가장 많은 분할 NFT가 거래되고 있는 플랫폼이다. DappRadar[22]에서 제공한 통계에 따르면 Doge NFT를 기반으로 Fractional.art에서 발행된 분할 NFT의 거래량이 $11.62M으로 가장 높게 나타났다.

Fractional.art에서 NFT를 소유하고 있는 사용자들은 해당 플랫폼을 통해 자신이 소유하고 있는 하나 혹은 여러 개의 NFT를 ERC-20, 혹은 ERC-1155 토큰으로 분할할 수 있다. 소유자는 자신의 NFT를 분할한 보상으로 정해진 기간마다 일정 양의 토큰을 추가로 지급받을 수 있다. 이렇게 발행된 FT들은 ERC-20을 기반으로 발행된 경우 다른 주소로 전송하거나 혹은 탈중앙화 거래소(Decentralized Exchange, DEX)[23]를 통해 유동성을 공급할 수 있다. ERC-1155를 기반으로 발행된 경우에는 OpenSea와 같은 다른 거래소에서 토큰을 판매하는 것도 가능하다.

한편, 누군가가 NFT 자체를 얻고자 할 때에는 두 가지 방법이 존재한다. 첫 번째는 해당 NFT를 기반으로 발행된 모든 분할 토큰을 소유하는 것이다. 이 경우 스마트 컨트랙트를 통해 자신이 보유한 모든 분할 토큰을 소각 하고 NFT를 자신의 지갑으로 전송시키는 것이 가능하다.

두 번째 방법은 경매를 통해 NFT를 구매하는 것이다. 분할된 조각을 구매한 소유자들은 플랫폼을 통해 자신의 판매 가격을 설정할 수 있다. 전체 50% 이상의 조각에 판매 가격이 설정된 경우 각 판매가격을 기반으로 평균치를 내어 NFT의 가격이 설정된다. NFT를 구매하고자 할 경우 해당 가격 이상의 금액을 스마트 컨트랙트에 입찰하여야 한다. 입찰을 하고나면 일정 기간 동안 경매가 시작되고 경매 기간 내에 가장 많은 금액을 입찰한 사람이 해당 NFT를 소유하게 된다. 이 경우에는 분할 조각이 여전히 조각 구매자들에게 속해있기 때문에 이를 일괄적으로 소각할 수 없으며 조각에 따른 정산 역시 자동적으로 진행하기 어렵다. 따라서 조각 소유자들이 스마트 컨트랙트를 통해 자신의 조각을 소각하고 자신이 보유했던 지분만큼의 금액을 지급받게 된다.

4.2. NFTFY

NFTFY[24] 또한 NFT를 분할하여 거래할 수 있게 한 플랫폼이나 Fractional.art와 달리 ERC-1155로의 분할은 지원하지 않는다. 해당 플랫폼에서는 사용자들이 자신이 보유한 NFT를 1,000,000개의 ERC-20 토큰으로 분할할 수 있다. 이때, NFT 소유자는 미리 NFT 판매 가격을 설정할 수 있다. NFT를 구매하고자 한다면 모든 조각을 구매하거나

표 4. 분할 NFT 플랫폼
Table 4. Fractional NFT Platforms

	Fractional.art	NFTFY	Unic.ly	Otis
NFT	ERC-721, ERC-1155	ERC-721, ERC-1155	ERC-721, ERC-1155	ERC-721
FT	ERC-20, ERC-1155	ERC-20	ERC-20	비공개
자산 종류	디지털 자산	디지털 자산	디지털 자산	디지털 자산 + 실물자산
NFT 구매 방식	가격 지불 + 경매	가격 지불	가격 지불 + 경매	가격 지불 + 경매
활용 블록체인	이더리움	이더리움, 폴리곤	이더리움	비공개

NFT 가격에 해당하는 금액을 지불함으로써 구매를 진행할 수 있다. 모든 조각을 구매한 경우에는 이를 NFT로 교환하는 것이 가능하고 NFT를 바로 구매한 경우 조각 보유자들은 자신이 보유한 지분에 따른 보상을 요청할 수 있다.

4.3. Unic.ly

Unic.ly[25]에서는 ERC-721 또는 ERC-1155로 발행된 NFT를 ERC-20의 FT로 분할할 수 있도록 지원한다. 하나의 스마트 컨트랙트 내에 여러 개의 NFT가 저장될 수 있으며 상황에 따라 소유자가 NFT를 추가할 수 있고 구매자가 여러 개의 NFT들 중 일부를 구매하고자 할 경우 따로 구매하는 것도 가능하다. 다른 플랫폼들에서는 NFT가 최종적으로 판매되었을 때 조각 토큰 보유자들이 자신의 보유 조각을 소각하고 지분에 따른 보상을 받았던 것과 달리 Unic.ly에서는 조각 보유자들이 항상 자신의 조각 토큰을 소각하고 지분에 따라 스마트 컨트랙트에 입금된 ETH을 얻을 수 있다.

처음 NFT 소유자가 자신의 NFT를 분할할 경우 생성된 ERC-20 토큰들을 유동성 풀에 추가하여 자유롭게 거래되도록 할 수 있다. 또한, 자신이 분할하고자 한 각 NFT 마다 가격을 설정하여 누구나 설정된 가격을 지불하고 NFT를 소유하게 할 수 있다. 이 경우 일정 기간 동안 경매가 진행되며 경매 중 가장 높은 가격을 입찰한 사람이 NFT를 소유하게 된다.

4.4. Otis

앞선 플랫폼들이 디지털 자산을 기반으로 한 NFT를 다루며 웹에서 동작했던 것과 달리 Otis[26]는 실물 자산을 기반으로 한 NFT를 주로 취급하며 모바일 기기에서 동작하는 어플리케이션이다. Otis는 희귀한 수집품이나 미술품 등에 분할 NFT를 사용하여 손쉽게 투자할 수 있도록 하였다. 실물 자산은 디지털 자산과 달리 실제 보관할 장소를 필요로 하기 때문에 Otis에서는 별도의 금고에 실물 자산을 보관한다. 실물 자산 보유자가 자신의 자산을 등록하고자 할 경우 Otis를 통해 자산을 검증하고 보관을 맡기게 된다. 검증이 완료되면 Otis는 해당 자산에 대한 ERC-721 기반의 NFT를 발행하여 소유자에게 전달한다. 소유자는 해당 NFT의 판매 방식을 설정하여 구매자가 경매를 통해 구입하거나 자신이 정한 금액을 지불하여 구매 가능하도록 할 수 있다. Otis에서 발행한 NFT 역시 조각으로 판매되어 사용자들이 자산에 대한 분할 소유권을 소유할 수 있도록 하였다. 하지만 구체적으로 NFT를 분할하는 방법에 대해서는 공개되어 있지 않았다.

Ⅴ. 결론 및 향후 연구

본 논문에서는 분할 NFT에 대하여 소개하고 분할 NFT의 발행 과정을 이더리움의 ERC 표준에 따라 분석하였다. NFT와 FT를 어떠한 표준으로 발행하느냐에 따라 구현 방식 및 장단점이 다를 수 있기 때문에 분할 NFT를 발행 하는 과정에서 많은 분석이 필요하다. 본 논문에서는 분할 NFT의 활용 방안에 대해서도 소개하였다. 분할 NFT는 고가의 NFT를 여러 개의 조각으로 나누어 적은 비용으로 구매할 수 있게 만들기 때문에 투자자의 입장에서 접근성을 높이는 효과를 가져다주며 거래량의 증가로 상품의 유동성도 증가시킨다. 이러한 특성을 활용하여 여러 플랫폼들에서 분할 NFT를 발행 및 거래할 수 있도록 기능을 제공하고 있었고 디지털 자산뿐만 아니라 실물 자산에서도 분할 NFT가 활용되고 있었다.

향후 연구에서는 본 연구에서 분석한 분할 NFT 구현 방식들에 따른 구체적인 장단점에 대해 분석할 계획이다. NFT와 FT의 표준을 달리하여 4가지 구현 방식에 따라 거래 비용 및 수수료를 비교하고 분할 NFT를 활용하는 방식에 따라 어떠한 방식을 채용하였을 때 가장 효과적인지 비교하는 연구를 진행할 계획이다.

References

[1] Wang, Qin, et al. "Non-fungible token (NFT): Overview, evaluation, opportunities and challenges." arXiv preprint arXiv:2105.07447 (2021).

[2] Evans, Tonya M. "Cryptokitties, cryptography, and copyright." AIPLA QJ 47 (2019): 219.

[3] Coinmarketcap, Retrieved Nov. 26, 2022, from https://coinmarketcap.com/

[4] Cryptoslam. Retrieved Nov. 26, 2022, from https://cryptoslam.io/

[5] Wood, Gavin. "Ethereum: A secure decentralised generalised transaction ledger." Ethereum project yellow paper 151.2014 (2014): 1-32.

[6] Ronin Litepaper. Retrieved Nov. 26, 2022, from https://litepaper.roninchain.com/

[7] Yakovenko, Anatoly. "Solana: A new architecture for a high performance blockchain v0. 8.13." Whitepaper (2018).

[8] Hentschel, Alexander, et al. "Flow: Separating Consensus and Compute--Execution Verification." arXiv preprint arXiv:1909.05832 (2019).

[9] Axie Infinity Whitepaper. Retrieved Nov. 26, 2022, from https://whitepaper.axieinfinity.com/

[10] Suchow, Jordan W., and Vahid Ashrafimoghari. "The Paradox of Learning Categories From Rare Examples: A Case Study of NFTs & the Bored Ape Yacht Club." Available at SSRN 4082221 (2022).

[11] Schaar, Luisa, and Stylianos Kampakis. "Non-fungible tokens as an alternative investment: Evidence from cryptopunks." The Journal of The British Blockchain Association (2022): 31949.

[12] Sommer, Thorsten, et al. "Request for comments: Proposal of a blockchain for the automatic management and acceptance of student achievements." arXiv preprint arXiv:1806.09335 (2018).

[13] Nakamoto, Satoshi. "Bitcoin: A peer-to-peer electronic cash system." Decentralized Business Review (2008): 21260.

[14] Hirai, Yoichi. "Defining the ethereum virtual machine for interactive theorem provers." International Conference on Financial Cryptography and Data Security. Springer, Cham, 2017.

[15] Dannen, Chris. Introducing Ethereum and solidity. Vol. 1. Berkeley: Apress, 2017.

[16] Ethereum Improvement Proposals. Retrieved Nov. 26, 2022, from https://eips.ethereum.org/

[17] EIP-20: Token Standard. Retrieved Nov. 26, 2022, from https://eips.ethereum.org/EIPS/eip-20

[18] EIP-721: Non-Fungible Token Standard. Retrieved Nov. 26, 2022, from https://eips.ethereum.org/EIPS/eip-721

[19] EIP-1155: Multi Token Standard. Retrieved Nov. 26, 2022, https://eips.ethereum.org/EIPS/eip-1155

[20] OpenSea. Retrieved Nov. 26, 2022, https://opensea.io/

[21] Fractional.art. Retrieved Nov. 26, 2022, https://fractional.art/

[22] DappRadar. Retrieved Nov. 26, 2022, https://dappradar.com/

[23] Malamud, Semyon, and Marzena Rostek. "Decentralized exchange." American Economic Review 107.11 (2017): 3320-62.

[24] NFTFY. Retrieved Nov. 26, 2022, https://www.nftfy.org/

[25] Unicly Litepaper. Retrieved Nov. 26, 2022, https://docs.unic.ly/

[26] Otis. Retrieved Nov. 26, 2022, https://www.withotis.com/

최 원 석 (Won-seok Choi)

2020~현재 포항공과대학교 컴퓨터공학과 통합과정

2020 포항공과대학교 컴퓨터공학과 학사

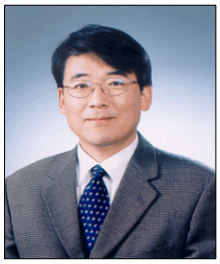
<관심분야> 블록체인, 암호화폐

우 종 수 (Jong-soo Woo)

2020~현재 포항공과대학교 정보통신대학원 연구교수
2020~현재 포스텍 크립토블록체인연구센터 공동센터장
2018~현재 DGIST 이사장
2016~2019 포스코교육재단 이사장
2014~2016 산업과학기술연구원 원장
2011~2014 포스코 기술연구원 원장

<관심분야> 암호화폐, 블록체인

홍 원 기 (James Won-Ki Hong)



2020~현재 포스텍 크립토블록체인연구센터 공동센터장
2012~2014 KT 종합기술원, 원장
2008~2012 포항공과대학교, 정보전자융합공학부장
2008~2010 포항공과대학교, 컴퓨터공학과 주임교수
2007~2010 포항공과대학교, 정보통신연구소 연구소장
2007~2011 포항공과대학교, 정보통신대학원장
1995~현재 포항공과대학교, 컴퓨터공학과 교수
1992~1995 Univ. of Western Ontario, 연구교수

<관심분야> 네트워크 트래픽 모니터링, 네트워크 및 시스템 관리, SDN/NFV, IoT, 무크기반 교육, 블록체인 및 암호화폐, Vmeeting 화상회의시스템

KNOM Review '22-02 Vol.25 No.02
https://doi.org/10.22670/knom.2022.25.2.30

GPU 데이터 캐시 접근 패턴 방법에 따른 성능 변화 분석

테오도라 아두푸◆, 김 윤 희°

A Performance Benchmark of Cached Data Access Patterns on GPUs

Theodora Adufu◆, Yoonhee Kim°

요 약

GPU는 다른 도메인의 응용 프로그램에 훨씬 더 높은 명령 처리량과 메모리 대역폭을 제공하므로 최근 범용 응용 프로그램을 성공적으로 가속화했다. 그러나 큰 메모리 대기 시간으로 인해 GPU 성능에 병목 현상이 남아 있습니다. 캐시는 칩 외부 메모리 트래픽을 줄이지 만 캐시 관리는 어렵다. Nvidia Ampere 아키텍처에 도입된 새로운 상주 제어 기능을 통해 사용자는 이제 캐시에 상주하는 데이터의 양을 제어할 수 있다. 그러나 여러 애플리케이션을 동시에 실행하는 동안 주의할 점은 데이터 지속성이 필요한 애플리케이션과 그 양을 식별하는 것이다. 이 백서에서는 처리량 및 데이터 액세스 빈도로 워크로드를 특성화하고 세 가지 공동 스케줄링 시나리오를 실험하여 최적의 성능을 위한 영구 캐시 할당을 결정한다. 서로 다른 데이터 액세스가 있는 응용 프로그램을 함께 실행할 때 L2 별도 할당이 지속적인 데이터 액세스가 있는 응용 프로그램에 편향되어서는 안 된다는 것을 관찰했다.

Key Words : L2 cache, Memory Access Pattern, Set-aside aware, GPU, L2 Access Cache Window Policy

ABSTRACT

GPUs have successfully accelerated general-purpose applications in recent times as they provide much higher instruction throughput and memory bandwidth to applications from different domains. There remains however a bottleneck in the performance of GPUs due to large memory latencies. Caches reduce off-chip memory traffic however, managing caches is difficult. With the new residency control feature introduced in the Nvidia Ampere architectures, users can now control how much data is resident in the cache. During co-executions of multiple applications, the caveat however is to identify which application requires data persistence and by how much. In this paper, we characterize workloads by throughput and data access frequencies and experiment with three co-scheduling scenarios to determine persistent cache allocations for optimum performance. We observed that when co-executing applications with different data accesses, L2 set-aside allocations should not be biased towards applications with persistent data accesses.

※이 성과는 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2021R1A2C1003379).

◆ First Author : Sookmyung Women's University Department of Computer Science, theoadufu@sookmyung.ac.kr
° Corresponding Author: Sookmyung Women's University Department of Computer Science, yulan@sookmyung.ac.kr

논문번호 : KNOM2022-02-04, Received December 1, 2022; Revised December 10, 2022; Accepted December 16, 2022

Ⅰ. Introduction

Graphics Processing Units (GPUs) in comparison to traditional CPUs, provide much higher instruction throughput and memory bandwidth during the execution of applications [1]. Since the introduction of NVIDIA's Compute Unified Device Architecture (CUDA), many applications including High Performance Computing (HPC) scientific applications [2][3][4][5][6][7][8][9], have leveraged the higher capabilities to run faster on the GPU. As researchers and private data centers have increased their deployments of applications on GPUs, shared clouds like Microsoft Azure[10], Amazon EC2[11] also provide GPU-based infrastructure services to support GPU clouds. There remains however, a bottleneck in the performance of the GPU. GPUs can hide memory access latencies with computation however, misaligned memory accesses, poor data locality in the cache memory, cache thrashing, high miss rates and poor thread and block size configurations can have an expensive impact on performance. HPC applications for instance, seek to exploit more parallelism through the use of multiple threads, however these active threads contend for limited GPU cache resources during execution. This results in cache thrashing and high miss rates. Memory optimization is thus, the most important area for performance improvement. This paper investigates the use of L2 cache residency control as a means of optimizing memory and thus improving performance.

● We characterize applications by their data request sizes and by the frequency of data reuse.
● We apply residency control to optimize memory and improve performance.
● We investigate the impact of the size of the L2 set-aside cache area on the performance of applications when run concurrently.

In summary, this is a quantitative study on exploiting data access frequency in the L2 cache for a set-aside-aware execution of applications which improves performance.

The paper is organized as follows: in Section 2, we briefly discuss some related works on optimizing L2 cache performance. We give a background of the heterogeneous memory system of the GPU architecture and the L2 Cache residency control feature in Section 3. We explain our experimental setup in Section 4 and present the quantitative results in Section 5. We conclude the paper in Section 6.

Ⅱ. Related Works

Multiple memory optimization techniques and approaches have been employed to mitigate the effect of memory limitations. Though there has been several studies to exploit the data locality in GPUs [12] [13], Sohan Lal et al.[14] argue that there is a lack of quantitative analysis of data locality in GPUs.

2.1. Thrashing Improvement Techniques

When applied to GPUs, cache bypassing proposed in CPUs as a thrashing-resistant technique against early eviction in cachelines may not achieve the expected improvement[15]. Cache-conscious wavefront scheduling (CCWS) [13] improves the L1 cache hit rate in GPUs by alleviating inter-warp contention. These techniques however do not consider the size of reused data to improve performance.

2.2. Sectored Caches

Recent GPU architectures including the Nvidia Ampere architecture employ the use of sectored caches to fetch only the sectors that are requested instead of always fetching all the sectors of a cache line. In A30, a sectored-cache has a cache line size of 128 bytes (B), divided into four sectors. On a miss, a sector-cache will only fetch the 32 B sectors that are requested. A full cache

line is not automatically fetched however if all four sectors are requested, it is possible to fetch a full cache line. This leads to improved bandwidth utilization even for strided acceses. Jia. et al [12] explore the use of sectored caches to tackle issues such as data over-fetch and hence improve performance.

2.3. Residency Control

Walden et al [16] in an effort to maximize memory bandwidth utilization for a sparse linear algebra kernel explore the L2 residency control and the asynchronous copy instruction features of the A100 architecture. From their experiments, the use of L2 persistence and asynchronous memory copies improve the overall performance by 81.2%, which is slightly better than the original mapping algorithm with the new A100 features. They however did not explore the impact of the size of the L2 set-aside area on performance.

Ⅲ. GPU Memory Architecture

The GPU contains multiple small hardware units called Streaming Multiprocessors (SMs), on-chip L2 cache and a high bandwidth DRAM also known as the global memory (Fig. 1).

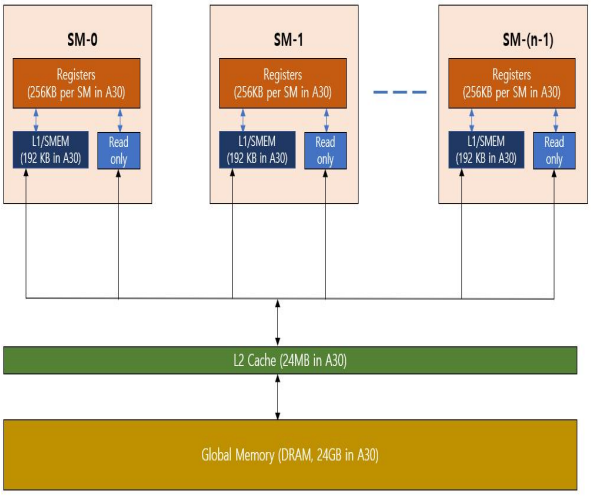


Fig. 1. Memory architecture of A30

The SMs can execute many threads concurrently. These threads are grouped physically into warps of 32 threads each. As stated GPUs contains many SMs, and these SMs can execute many threads concurrently. The threads in the SMs access data and instructions from the global memory (DRAM) at a given bandwidth. Bandwidth is best served by using as much fast memory and as little slow-access memory as possible. Thus, there exists on-chip memory such as registers which are allocated to individual threads, Read-only memory for specific tasks such as texture memory, and the L1 cache/shared memory, for fast data access within each SM.

The L1 cache/shared memory is on-chip memory that is shared within thread blocks or CUDA blocks. The shared memory usage is however controlled via software while L1 cache is controlled by hardware. Because L1 cache and shared memory exists on-chip, it is faster than both L2 cache and global memory. The L1 cache is however very small in size and not coherent. To ensure coherence in data accessed from the global memory by different SMs, the L2 cache is used.

L2 cache can be accessed by all threads in all CUDA blocks. Retrieving data from the L2 cache is faster than retrieving data directly from global memory (DRAM). In modern GPU architectures, access to global memory is cached in L1 and L2 by default.

3.1. L2 Caching Policy

The L2 cache potentially provides higher bandwidth and lower latency accesses to global memory. A typical cache line size is 128B in GPUs. In the Ampere architecture, the loads and stores can be serviced at 32 B granularity known as a sector[14].

When a CUDA kernel accesses a data region in the global memory repeatedly, such data accesses can be considered to be persisting. On the other hand, if the data is only accessed once, such data accesses can be considered to be streaming.

A caching policy is used to determine which portion of data to cache. Due to the relatively small ratio of cache sizes to input data sizes of many HPC applications, there is the need for a policy to select the data to prioritize for caching. Consequently, the data access frequencies are considered in the eviction of the data cache lines. Streaming data for instance is first in the eviction priority order and will likely be evicted when cache eviction is required. This policy is known as the evict_first policy.

Data considered as persistent and hence stored in the set-aside region will be last in the eviction priority order and will likely be evicted only after other data with the evict_first and evict_normal policies are evicted.

The persistence offered by the evict_first policy provides the opportunity to cache frequently accessed data and thus minimize the time spent in fetching newer cache lines from the global memory during executions.

3.2. L2 Cache Data Persistence Control

The A30 architecture offers a new feature that allows a portion of the L2 cache to perform persistent data accesses to device memory, which ultimately enables higher bandwidth and lower latency accesses to device memory.This is achieved through the use of APIs offered in the CUDA version 11 toolkit to set aside a portion of the 24-MB L2 cache to perform persistent data accesses to global memory. If this set-aside portion is not used by persistent accesses, normal accesses or streaming data can use it.

The L2 cache set-aside size for persisting accesses may be adjusted, within limits. For our experiments, we set aside a limit of 75% of the L2 cache memory for persisting accesses. Since the L2 cache size of the A30 GPU is 24MB, this translates to 18MB of L2 cache memory set aside for persistent accesses.

Ⅳ. Experimental Setup

4.1. Hardware and Software Description

Table [1] summarizes the experimental setup for our experiments. We execute our experiments on an Nvidia Ampere GPU device with 24GB of Device memory and 24MB of L2 cache. We use 1 of the 2 GPUs in our A30 environment. The compute capability is 8.0 and a cache line has a size of 128 Bytes. For profiling, we use Nvidia's Nsight Compute [17].

Table 1. Experimental Setup

GPU Device	NVIDIA A30
Compute capability	8.0
Device Memory	24GB
GPU memory bandwidth	933 GB/s
L2 cache size	24MB
L1 cache size	192KB
Profiler	Nsight Compute

4.2. Sliding Window Experiment

We implement a sample micro-benchmark [18] which uses a 1024 MB region in GPU global memory through the following kernel code Fig. 2.

```
__global__ void kernel(int *data_persistent, int *data_streaming, intdataSize, int freqSize)
{
int tid = blockIdx.x * blockDim.x + threadIdx.x;

data_persistent[tid % freqSize] = 2 * data_persistent[tid % freqSize];data_streaming[tid %
dataSize] = 2 * data_streaming[tid % dataSize];

}
```

Fig. 2. Kernel code for sliding window experiment [18]

An access policy window Fig. 3 specifies a contiguous region of global memory and a persistence property in the L2 cache for accesses within that region. As shown in Fig. 3, the stream level attribute data structure is used to set the region of the device memory which will persist in L2 cache when initially accessed.

```
cudaStreamAttrValue stream_attribute;                                    // Stream level attributes data structure
stream_attribute.accessPolicyWindow.base_ptr  = reinterpret_cast<void*>(ptr);  // Global Memory data pointer
stream_attribute.accessPolicyWindow.num_bytes = num_bytes;               // Number of bytes for persisting accesses.
                                                                         // (Must be less than Max Window Size)
stream_attribute.accessPolicyWindow.hitRatio  = 1.0;                     // Hint for persisting accesses in the num_bytes
region
stream_attribute.accessPolicyWindow.hitProp   = cudaAccessPropertyPersisting;  // Type of access property on cache hit
stream_attribute.accessPolicyWindow.missProp  = cudaAccessPropertyStreaming;  // Type of access property on cache miss.

//Set the attributes to a CUDA stream of type cudaStream_t
cudaStreamSetAttribute(stream, cudaStreamAttributeAccessPolicyWindow, &stream_attribute);
```

Fig. 3. Access Policy for sliding window experiment [18]

Similar to NVIDIA's sliding window experiment [18], we specified the access to the freqSize * sizeof(int) bytes of memory to be persistent and varied this persistent data region from 10MB to 40MB (Fig. 4) to model various scenarios where data fits in or exceeds the available set-aside portion of 18MB for our NVIDIA A30 GPU. Normal or streaming accesses can use the remaining 6MB of the non set-aside L2 portion. We used a fixed hit-ratio of 1.0 for our experiment.

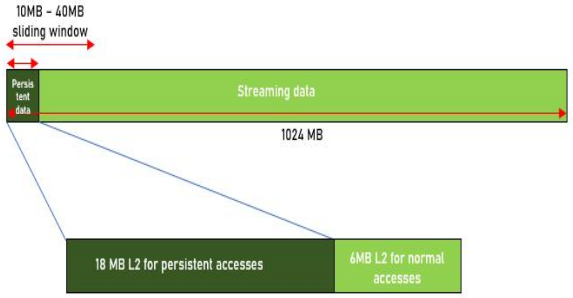


Fig. 4. Mapping the persistent accesses to L2 set-aside for our Nvidia A30 environment

The performance of the kernel in Fig. 2 is shown in Fig. 5 and Fig. 6. With a hit ratio of 1.0, the hardware attempted to cache the whole 40MB window in the set-aside L2 cache area. However, since the set-aside area (18MB) was smaller than the window, cache lines were evicted to make room for data required for the executions. The premature eviction of data before any significant reuse is known as cache thrashing.

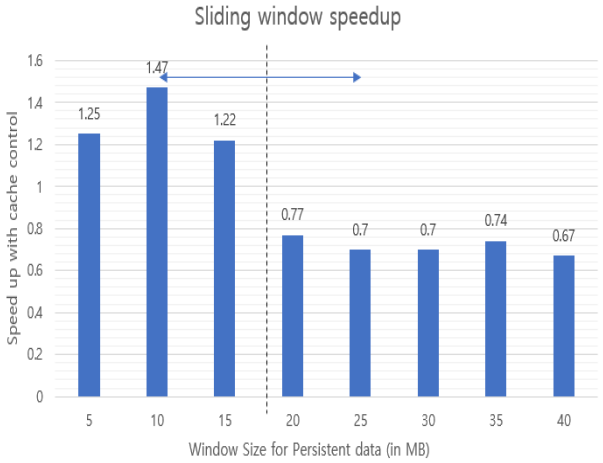


Fig. 5. Sliding window with hit-ratio

When the persistent data region fitted well into the 18MB set-aside portion of the L2 cache, a performance increase of as much as 47% is observed in Fig. 5. However, once the size of this persistent data region exceeded the size of the L2 set-aside cache portion, there was approximately 25% drop in performance. We attribute the fall in performance to thrashing of L2 cache lines.

Controlling the hit-ratio

The hitRatio value can be used to manually control the amount of data different accessPolicyWindows from concurrent CUDA streams can cache in L2. The hitRatio can therefore be used to reduce the amount of data moved into and out of the L2 cache.

In order to optimize the performance and reduce thrashing, we tuned the numbytes and hitratio parameters in the access window so that a random 10MB of the total persistent data (data_in_cache) was resident in the L2 set-aside cache portion and investigated the performance with an experiment. According to equation 1, this translated to a hit ratio of 0.556.

$$Hitratio = \frac{data\_in\_cache}{numbytes} \qquad (1)$$

From the results in Fig. 6, we observed an

overall improvement in performance for L2 set-aside cache portions which exceeded the size of the persistent data region.

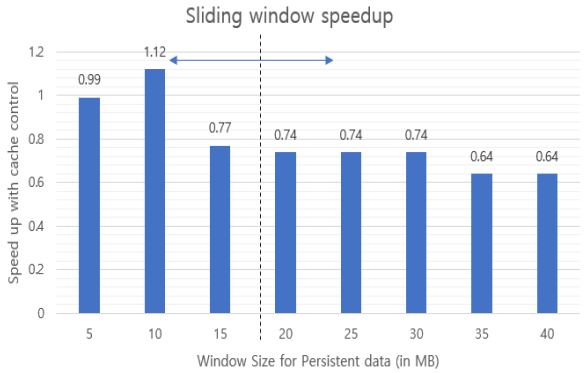


Fig. 6. Sliding window with adjusted hit-ratio

4.3. Workloads

We chose two workloads from the cudaSDKsamples[19] benchmark suites with data input sizes greater than our L2 cache size for our experiments. A short description of the workloads, the input data sizes and the kernels per workload is given below and summarized in Table 2.

Table 2. Description of workloads

Workload	Kernels and Functions	Input Data, MB
Histogram	Histogram64, mergedHistogram64	64
Conjugate Gradient	gpuConjugateGradient	108.8

Histogram: A histogram is a commonly used analysis tool in image processing and data mining applications. They show the frequency of occurrence of each data element [20]. The histogram used in our experiments has two (2) kernels: histogram64() and mergedhistogram64() used for analysis in our experiments.

Conjugate Gradient (CG) Solver: The conjugateGradientMultiBlockCG implements a conjugate gradient solver on a GPU using Multi Block Cooperative Groups. The sample used in our experiments has only one kernel [21].

4.4. Application Characterization for Frequent Accesses

We began by profiling each of the kernels for the workloads to determine the L2 cache access patterns. This was to inform the decision on the size of cache resources to be allocated to each application during scheduling. We considered the approach by Alsop et al. [22] and characterized the kernels according to the following:

Memory intensiveness: As a general rule, workloads with low compute bandwidth and high memory request bandwidth are more likely to be sensitive to caching policy than workloads with low memory request bandwidth and high compute bandwidth [22].

Frequent Data Accesses: Kernels with smaller data sizes generated from global memory into L2 cache but with relative large data sizes generated to L1 cache can be considered to be reusing data in the L2 cache compared to other kernels and are thus classified to have Persistent accesses. Kernels with similar or same data size generated from global memory into L2 cache and into L1 cache can be considered to have Streaming accesses. Finally, kernels considered to have Normal accesses are those with relatively smaller sized data generated into L1 cache compared to data generated from global memory into L2 cache.

V. Results

5.1. Application Characterization

Based on the values of the compute throughput and memory throughput for each of the kernels shown in Fig. 7, we ascertained that all the kernels were memory intensive with kernel histogram64 being the most memory intensive.

We also collected metrics on data request and access sizes to obtain insight into which

workloads have persistent accesses and which

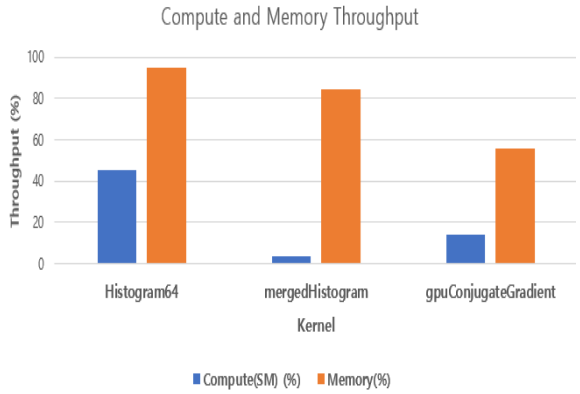


Fig. 7. The compute and memory throughput for the kernels

ones have streaming accesses. Based on the difference in the size of data generated in L1 relative to the data cached, we identified the access type for each kernels as either persistent, streaming or normal according to the caching policy used for Nvidia Ampere devices.

Table 3. Data transfer by the L2 cache

Kernel	Data Request MB	Data cache MB	Data to L1 MB	Access type
Histogram64	64	64	100	Streaming
mergedHistogram64	136.56	1.14	143.20	Persistent
gpuConjugateGradient	108.8	40.39	46.82	Normal

From Table 3, mergedHistogram64 was characterized to have persistent access type as it accessed the L2 cache more frequently. The data transferred through the L2 cache is also represented graphically in Fig. 8.

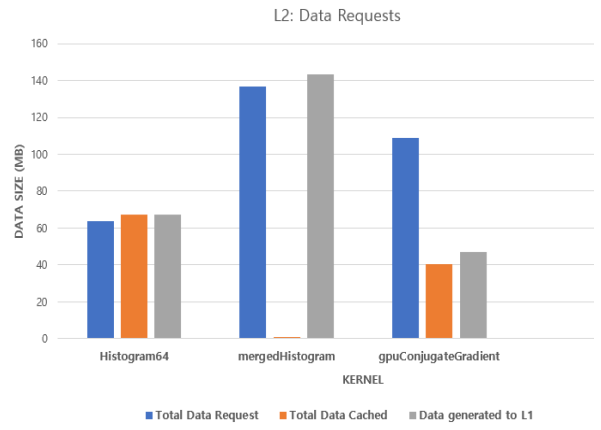


Fig. 8. Data transferred by the L2 cache

5.2. Warp Stalls

The warp scheduler can mask the delay of the warp by switching to a different warp when one warp is stopped owing to memory work or other reasons. With the Nsight Compute profiler, we collected warp stall sampling metrics for the first 100 address spaces during the execution of histogram and conjugated gradient kernels We compared the effect of allocating set-aside area to one of the kernels at a time during concurrent executions.

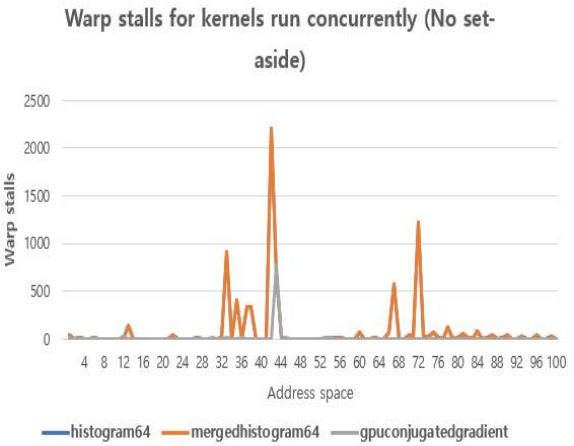


Fig. 9. Warp stalls for kernels run concurrently (no set-aside)

From the results in Fig. 9, we observed that, when there was no set-aside area in the L2 cache, histogram64 kernel did not have a warp stall. This confirmed the assertion that the data loaded into the L2 cache is hardly re-used.

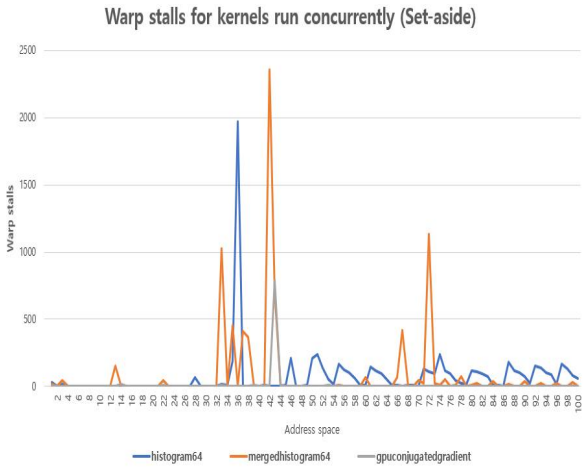


Fig. 10. Warp stalls for kernels run concurrently (set-aside)

On the other hand, when a set-aside area of 18MB is allocated to histogram64, the kernel had a bursty warp stall (Fig. 10).

NVIDIA maintains that normal or streaming accesses can use set-aside portions of the L2 when not in use however this may come at a cost of memory stalls.

5.3. Set-aside aware concurrent executions

The above observation for warp stalls revealed the need to identify the portion of L2 cache to allocate for persistence during concurrent executions in order to maximize the overall performance of the applications.

To investigate this, we considered three co-scheduling scenarios showing different allocations of L2 cache.

We set the hit ratio to 1.0 and the maximum persistence of the L2 cache to 0.75% of the total size of the L2 cache.

At the given hit ratio of 1.0, we tuned the window size of the applications for each concurrent run as follows: (H/mH=3, CG=15), (H/mH=9, CG=9) and (H/mH=15, CG=3). We observed the differentials in the number of elapsed cycles as shown in Table 4.

Table 4. Differentials in Elapsed Cycles for different set-aside areas

Kernel	H/mH=3 CG=15	H/mH=9 CG=9	H/mH=15 CG=3
Histogram64, H	−11312	16072	41104
mergedHistogram, mH	−12698	57302	−48538
gpuConjugateGradient, CG	51170	94080	115122

From the results, we observed that, because of the streaming nature of data accesses in histogram64, there was poor performance when 15MB of the L2 cache was reserved for persistent access by gpuConjugateGradient as the Histogram64 kernel had more elapsed cycles of (11312 cycles) relative to execution in single-mode.

On the other hand, there was a general increase in performance when the larger persistent data region (15MB) was allocated maximally to the histogram; the Histogram64 kernel had fewer elapsed cycles of about 41104 cycles relative to execution in single-mode.

The normalized speed-up values for the concurrent executions of kernels in both the histogram and conjugate gradient workloads according to the folllowing set-aside allocation (H/mH=3, CG=15), (H/mH=9, CG=9) and (H/mH=15, CG=3) is represented in Fig. 11.

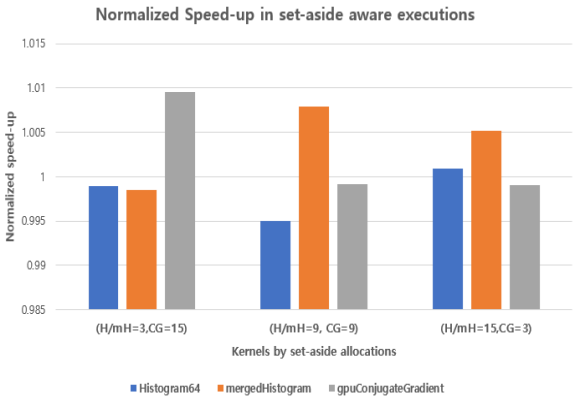


Fig. 11. Normalized speed-up in set-aside concurrent executions

From the results, we observed that, for optimal performance of all applications, larger persistent set-aside area must be allocated to kernels with streaming accesses such as histogram64, to enhance overall performance.

VI. Conclusions and Future Works

According to Alsop J. et al [22], although caching can significantly improve performance by enabling local data reuse, in some cases the best caching policy is not the one that enables the most caching.

With the new residency control feature introduced in Nvidia Ampere architectures, an end-user must decide on the best caching policy by identifying the access patterns of the workloads.

The kernel that generates the most data from the global memory may not necessarily be the kernel which requires persistence in the L2 cache. Contrary to the intuition to allocate less persistent data region to a kernel with streaming access whilst concurrently allocating more to the kernel with either normal or persistent data access, we observed that, allocating more persistent region to a kernel with streaming access when co-executed with that of normal access yielded optimal overall performance.

In future, we intend to expand the number of workloads profiled and observe their behavior for different co-scheduling scenarios.

References

[1] Nvidia Programming Guide, https://docs.nvidia.com/cuda/cuda-c-programmingguide/index.html

[2] Tensorflow, https://www.tensorflow.org/tutorials/using_gpu

[3] LAMMPS, http://lammps.sandia.gov/doc/accelerate_gpu.html

[4] NCBI-BLAST, https://www.ncbi.nim.nih.gov/pmc/articles/PMC3018811

[5] GAUSSIAN, http://gaussian.com/relnotes/?tabid=2

[6] NAMD, http://www.ks.uiuc.edu/Research/namd/2.9/ug/node88.html

[7] VASP, http://www.vasp.at/index.php/news/44-administrative/115newreleasevasp541withgpusupport

[8] AMBER, http://ambermd.org/gpus/

[9] GAMESS, http://www.msg.ameslab.gov/gamess/versions.html

[10] Microsoft Azure, azure.microsoft.com/enus

[11] EC2 ELASTIC GPUS, https://aws.amazon.com/ec2/ElasticGPUs/,2017.

[12] Jia, W., Shaw, K.A., Martonosi, M., "Characterizing and improving the use of demand-fetched caches in GPUs", in proceedings of the 26th ACM international conference on supercomputing, ICS (2012)

[13] Rogers, T.G, O'Connor, M.and Aamodt, T.M., "Cache-conscious wavefront scheduling", Proceedings of the 45th annual IEEE/ACM International symposium on microarchitecture, MICRO (2012)

[14] Lal, S., Sharat Chandra Varma, B., and Juurlink, B. (2022), "A Quantitative Study of Locality in GPU Caches for Memory Divergent Workloads" International Journal of Parallel Programming, 50, 189216. https://doi.org/10.1007/s10766022007292

[15] Chen X., Chang L., Rodrigues C., Ly Jie., Wang Z. and Hwu W., (2014) "Adaptive Cache Management for Energy-Efficient GPU Computing" MICRO-47: Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture December 2014697 pages ISBN:9781479969982

[16] A. Walden, M. Zubair, C. P. Stone and E. J. Nielsen, "Memory Optimizations for Sparse Linear Algebra on GPU Hardware," 2021 IEEE/ACM Workshop on Memory Centric High Performance Computing (MCHPC), 2021, pp. 25-32, doi: 10.1109/MCHPC54807.2021.00010.

[17] NSIGHT COMPUTE, https://developer.nvidia.com/nsight-compute

[18] Sliding Window Experiment, https://docs.nvidia.com/cuda/cuda-c-best-practicesguide/index.htmlmemory-optimizations

[19] CUDA SAMPLES https://github.com/NVIDIA/cudasamples/

[20] Nvidia developer downloads, https://developer.download.nvidia.com/compute/cuda/1.1-Beta/x86_website/projects/histogram64/doc/histogram.pdf

[21] CUDA Samples Documentation, https://docs.nvidia.com/pdf/CUDA_Samples.pdf
[22] J. Alsop et al., “Optimizing GPU Cache Policies for MI Workloads,” 2019 IEEE International Symposium on Workload Characterization (IISWC), 2019, pp. 243-248, doi:10.1109/IISWC47752.2019.9041977.
[23] N. Duong et al., “Improving cache management policies using dynamic reuse distances”, in Proceedings of the 45th Annual IEEE/ACM International Symposium on Microarchitecture, 2012

테오도라 아두푸 (Theodora Adufu)

2013년 : 가나 대학교 컴퓨터과학 및 경제학과 졸업(학사)
2016년: 숙명여자대학교 컴퓨터과학과 졸업(석사)
2022년～현재: 숙명여자대학교 컴퓨터과학과 (박사)

<관심분야> 컨테이너, 클라우드 컴퓨팅, HPC 클라우드, Accelerated Computing(GPUs)

김 윤 희 (Yoonhee Kim)

1991년 : 숙명여자대학교 전산학과 졸업(학사)
1996년 : Syracuse University 전산학과 졸업(석사)
2003년: Syracuse University 전산학과 졸업(박사)
1991년～ 1994년 한국전자통신연구원 연구원.
2000년~2001년 Rochester Institute of Technology 컴퓨터공학과 조교수.
2001년 ～ 2016년 숙명여자대학교 컴퓨터과학부 교수.
2017년 ～ 현재 숙명여자대학교 소프트웨어학부 교수.

<관심분야> 클라우드 컴퓨팅, 워크플로우 제어, HPC 클라우드, Accelerated Computing(GPUs)

KNOM Review '22-02 Vol.25 No.02
https://doi.org/10.22670/knom.2022.25.2.40

경량화 데이터와 딥러닝 모델을 적용한 효율적인 네트워크 트래픽 분류 방법

신 창 의◆, 박 지 태*, 백 의 준*, 최 정 우*, 김 명 섭°

Efficient Network Traffic Classification Method Using Lightweight Data and Deep Learning Model

Chang-Yui Shin◆, Jee-Tae Park*, Ui-Jun Baek*, Jung-woo Choi*, Myung-Sup Kim°

요 약

트래픽 분류는 컴퓨터 네트워크 영역에서 서비스관리 및 보안 등의 분야에서 그 역할이 점점 더 중요해지고 있다. 초기에는 포트넘버, DPI, 통계정보 등을 활용해 트래픽 분류가 가능했다. 그러나 정보보호 측면에서 트래픽의 페이로드가 암호화되면서 분류가 제한되었지만 머신러닝기법이 추가 활용되면서 문제점이 해결됐다. 이후 딥러닝 모델들이 활용되고 성능은 향상되었으나, 많은 변수를 입력으로 넣어도 트래픽 분류가 가능해 짐에 따라 모델과 데이터가 점점 무거워져서 자원과 시간이 많이 소모되었다. 부담스럽고 성가시게 된 모델과 데이터 경량화 활용에 본 연구의 목적을 두고, KD(Knowledge distillation) 기법을 바탕으로 BERT가 경량화된 DistilBERT를 선정했고, 경량화한 데이터를 적용했다. 첫 번째 패킷 1개 중 앞 100bytes 크기의 입력데이터(패킷 단위)와 이러한 5개의 패킷이 연결된 입력데이터(플로우 단위)로 정확도 / F1-score가 각각 0.9707 / 0.9731 과 0.9703 / 0.9706로 매우 우수한 성능을 보였다.

Key Words : Encrypted Network Traffic Classification, BERT, Knowledge Distillation, Efficiency

ABSTRACT

Traffic classification is becoming more and more important in areas such as service management and security in the area of computer networks. In the early days, it was possible to classify traffic using port numbers, DPI, and statistical information. However, in terms of information protection, classification was limited as the traffic payload was encrypted, but the problem was solved by additionally using machine learning techniques. Since then, deep learning models have been used and performance has improved, but as traffic classification became possible even with a large number of features as inputs, models and data became increasingly heavy like the front and back of a coin, consuming a lot of resources and time. With the purpose of this study to lighten the burdensome and cumbersome model and data, DistilBERT selected the lightweight BERT based on the previously presented KD (Knowledge distillation) research and applied the lightened data. With a 100bytes input of the first packet(packet unit) and an input(flow unit) of these 5 packets, the accuracy / F1-score was 0.9707 / 0.9731 and 0.9703 / 0.9706, respectively, showing very good performance.

※ 이 논문은 2021년도 교육부의 재원으로 한국연구재단의 지원을 받아 수행된 지자체-대학 협력기반 지역혁신 사업의 결과(2021RIS-004)이고, 2020년도 산업통상자원부 및 한국산업기술평가관리원(KEIT) 연구비 지원에 의한 연구(No. 20008902, IT비용 최소화를 위한 5채널 탐지기술 기반 SaaS SW Management Platform(SMP) 개발) 결과다.
◆ First Author : Defense Agency for Technology and Quality, superego99@gmail.com
° Corresponding Author : Korea University, Department of Computer Convergence Software, tmskim@korea.ac.kr
* Korea University, Department of Computer Convergence Software, {pjj5846, pb1069, choigoya97}@korea.ac.kr
논문번호 : KNOM2022-02-08, Received December 2, 2022; Revised December 10, 2022; Accepted December 16, 2022

Ⅰ. 서 론

네트워크에 연결된 퍼스널 디바이스, 다양한 서비스용 장비, 클라우드 서버 등이 인터넷을 통해 상호 연결되어 데이터를 공유하여 SNS, 헬스, 교통, 에너지관리 등의 서비스 산업들이 경계구분 없이 급격히 성장하고 있다. 이러한 서비스들의 QoS, 지능네트워크 운영, 관리 및 보안 등을 보장하기 위한 좋은 해결책이 네트워크 트래픽 분류 및 식별인데, 이것은 컴퓨터 및 네트워크 기술의 발전과 함께 지속적으로 계속 연구되고 있다.

트래픽 클래스 분류는 통상 트래픽 타입, 프로토콜, 애플리케이션 등의 3가지[1]에 따라 이루어진다. 초기 단계의 트래픽 분류는 IANA에 등록된 전송계층의 포트번호에 크게 의존하는 것이 일반적이다. 물론 포트번호 정보 1가지로만 분류하는 것이 아니고, 패킷도착시간, 패킷길이 및 플로우 분포 등의 통계적 정보들도 부가적으로 사용되는 연구들이 줄을 이었다. 그 다음단계에서 진행되는 트래픽 분류의 흐름은 페이로드에서 특정 패턴이나 키워드를 찾아서 동일한 패킷들 간의 연관성을 통해서 분류하는 것[2]으로써 포트번호만 가지고 분류하는 것보다 높은 정확도를 보였다. 그도 그럴 것이 잘 알려진 포트를 사용하지 않는 애플리케이션들이 등장하게 된 탓이다.

그러나 이러한 Deep Packet Inspection(DPI) 기법도 페이로드가 암호화되면서 지속적으로 트래픽분류에 적용하기에 제한되었다. 그 다음 단계에서 대표적으로 등장한 것이 머신러닝기법[3]을 활용한 트래픽 분류였다. 트래픽 플로우의 처음 몇 패킷을 사용하고 Simple K-means에 기반해 서로 다른 TCP 기반 애플리케이션 프로그램을 분류하는 머신러닝기법의 접근[4]이 있었다. 웹 및 P2P 트래픽을 식별하기 위해 K-means를 기반으로 누락된 통계를 추정할 수 있는 머신러닝 기법의 접근[5]도 그 뒤를 이었다. 양방향 플로우 기반이든 단방향 플로우 기반이든 의미 있는 특징을 추출하여 K-Nearest Neighbors, Decision Tree, Naive Bayes 및 Support Vector Machine(SVM) 등의 머신러닝 기법의 알고리즘을 활용한 트래픽분류가 대세가 된 것이다.

그 이후에 Computer Vision(CV) 및 Natural Language Process(NLP)와 같은 분야에서 딥러닝의 활약에 따라 네트워크 트래픽 분류에서도 딥러닝이 등장하게 되었다. 딥러닝의 내부적인 발전 과정에 따라 수 많은 피처들 가운데 특정 피처에 대한 오버피팅을 방지하고, 학습 일반화를 하기 위해 Dropout, Batch nomalization 방법이 일반화되면서 보다 높은 성능을 나타내어 지속적으로 성장하였다. 초기에는 레이어를 쌓아서 인공신경망을 구성하는 Automatic Encoder(AE)[6]가 사용되기도 했으나, 딥러닝의 발전에 맞춰서 컴퓨터 비전분야에서의 대표적인 딥러닝기법인 Convolutional Neural Network(CNN)[7], 자연어처리 분야에서의 Recurrent Neural Network(RNN)[8]를 활용하는 연구들이 연이어 등장하면서 암호화된 트래픽 분류 정확도가 90%를 상회하는 높은 성능을 보이게 되었다. 이러한 가운데 앞서 시계열적 연속성이 있는 자연어처리에서 RNN보다 발전된 Transformer모델[9]이 등장하면서 딥러닝의 도약이 시작되었다. Transformer 모델은 자연어처리에서 뿐만이 아니라 다시금 컴퓨터 비전 분야에서까지 그 유용성이 확인되었다. 그럼에 따라 시계열적 연속성이 있는 패킷들로 구성된 네트워크 트래픽에도 적합할 것이라는 기대와 함께 네트워크 트래픽분류 분야에서도 이를 활용한 연구[10][11]들이 시작되었다.

본 논문에서는 Transformer를 기반으로 발전한 Bidirectional Encoder Representations from Transformers (BERT)모델[12]을 경량화한 DistilBERT모델[13]과 경량화한 데이터를 활용해 트래픽 분류에 적용했으며, 논문에서 제시하는 분류 방법이 가지는 주요한 연구성과는 다음의 3가지로 볼 수 있다.

1) 헤더정보와 암호화된 페이로드 일부 정보로 데이터를 경량화하고 모델에 반영해, 별도의 사전학습 과정 없이 짧은 파인튜닝시간 만으로 높은 분류성능을 나타내었다.
   트래픽 데이터의 첫 번째 패킷에서 100bytes (패킷 단위)와 양방향 패킷으로 구성된 플로우에서 앞 5개의 패킷 각각 100bytes만을 연결(플로우 단위)해 입력으로 사용했다. 6개 클래스의 데이터에서 패킷 단위 분류정확도는 0.9705, 플로우단위는 0.9703을 보였다. 20 에포크의 학습 시간이 1시간 40분이 채 걸리지 않았다.
2) 경량화한 트래픽 데이터에 맞춰서 분류에 사용하는 모델도 자연어처리분야의 기존 BERT 모델이 경량화된 DistilBERT모델을 활용해, 효율적인 트래픽 분류가 가능하도록 했다. 분

류하는데 소모되는 시간은 0.0099 second/packet임에 따라 패킷 단위 분류를 적용해 근실시간에 가까운 사용이 가능하며 하드웨어의 성능에 따라서 보다 향상될 수 있다.

3) Ground-truth 특성이 있는 공개 데이터 셋을 사용함으로써 실제 환경에서의 클래스 불균형 특성을 가진 데이터를 그대로 반영했다.

여기에 Class Weight를 적용함으로써 학습하는 과정이 안정적이고 빠르게 진행되도록 함으로써, 활용성 측면에서 클래스 불균형성이 있는 새로운 데이터에 비교적 빠르게 모델을 적용할 수 있도록 했다.

본 논문의 구성은 다음과 같다. 이어지는 제2장은 관련 연구와 연구 배경으로 세부 구분하여 구성되어 있다. 제3장은 데이터 전처리, 실험 준비 및 모델에 대한 설명으로 구성되어 있다. 제4장에는 모델을 활용한 트래픽분류 결과와 이에 대한 분석이 제시되어 있으며, 마지막으로 제5장에는 결론 및 향후 연구 방향이 기술되어 있다.

Ⅱ. 관련연구 및 연구배경

본 연구에서는 데이터와 딥러닝 모델 경량화에 중점을 두고 있는 만큼, 암호화된 트래픽분류와 관련된 딥러닝 기법 연구들에 대해 집중적으로 알아보았다. 이어서 데이터와 모델 각각에 대해 어떠한 배경으로 경량화 측면에서 연구가 시작되었는지 동기부여 관점 순서로 중점을 세부 구분해 살펴보았다.

2.1. 관련연구

2.1.1 CNN, RNN 모델

CNN과 RNN을 결합하여 조합된 딥러닝 모델이 네트워크 트래픽 분류에 적용되는 연구도 제시되었다. [7]에서 데이터로는 20개의 연속된 패킷으로 구성된 플로우를 기반으로 하는데, 각각의 패킷은 6개의 피처(source port, destination port, 페이로드 크기, TCP window size, 패킷 도착시간, 패킷 방향)로 구성된다.

모델 측면에서는, 시계열 특성을 보존한 데이터가 CNN모델을 거쳐서 특징 벡터가 생성하게 되는데, 이 특징 벡터가 Long Short-Term Memory (LSTM) 계층의 내부 히든레이어의 크기와 동일한 행렬벡터임에 따라 입·출력이 동일하게 적용되면서 Fully Connected(FC) layer로 연결되는 구조를 형성하였다. 이러한 구조를 통해 20개의 연속된 패킷을 분류에 적용해, 정확도 0.9632를 달성했다.

일반적으로 CNN 모델은 여러 개의 convolution layer, pooling layer 및 FC layer로 구성되어 있으며, 이전 레이어의 출력이 다음 레이어의 입력으로 사용된다. 여기서 출력 생성에 사용되는 커널로 동일한 크기를 적용함에 따라 학습가능한 파라미터의 수가 줄어들고, 이러한 특성에 따라 패턴을 캡처하는 커널은 위치에 관계 없이 패턴을 찾아낼 수 있다는 중요 점을 드러낸다.
이러한 점은 특정 패턴이 존재하고, 그 특정 패턴의 이동 불변성이 네트워크 트래픽에서도 마찬가지로 적용 가능한 것[14]으로 연구되었다.

2.1.2 Transformer 모델

네트워크 트래픽은 시계열적 특성을 가지고 있음에 따라 이를 시퀀셜한 구조를 가지는 RNN 모델을 적용하면 우수한 성능을 보였다.
그러나 주목받았던 RNN은 자체의 구조적 특징으로 인해 학습 간 획득한 attention weight이 학습진행에 따라 그 값이 손실되는 구조를 가진다는 단점이 제기됐다. 이에 positional encoding, self-attention 및 multi-head attention을 핵심으로 하는 Transformer 모델[9]이 대안으로 제시되었다.

가장 중요한 점은 positional encoding을 사용하여 입력된 데이터의 순서정보가 보존되는데, 같은 입력 값이 다른 위치에 입력되면 그 순서를 반영해 다른 임베딩 벡터값을 가지게 한다는 것이다.

self-attention은 자기 자신에게 주의 집중한다는 의미로서, positional encoding을 거쳐서 입력된 입력 데이터들 간에 유사도를 구함으로써 연관성을 찾아내는 것이다.

multi-head attention은 한 번만 어텐션 헤드를 사용 것 보다 여러 번 병렬로 사용되는 것이 효과적이라는 사실하에, 8개의 병렬 어텐션 행렬로부터 다양한 시각에서의 정보를 얻는다. 그리고 모든 어텐션 헤드를 연결하여 종합된 정보를 얻어내는 구조로 되어 있다. 이러한 Transformer 모델의 self-attention 구조로 패킷 하나만을 분류에 적용한 연구[8]에서 정확도 0.9480의 성능을 보였다.

2.1.3 BERT 모델

Transformer 구조를 기반으로 특화된 BERT는

그 이름에서도 알 수 있듯이 Transformer의 인코더를 활용해 입력 데이터를 양방향 관점에서 인식하는 것이다. BERT에는 입력으로 3개의 벡터들이 동시 들어가게 된다.

첫 번째는 각 문장이 토큰 단위로 구분되는 embedding 벡터이고, 두 번째는 Position에 대한 벡터이며, 마지막은 토큰이 어느 문장에 속하는지에 대한 segmentation 벡터이다.

이렇게 입력된 3개의 벡터를 두 가지 방법으로 사전학습 하는 메커니즘을 가지고 있다.
하나는 입력 데이터를 15%의 확률로 단어를 선택하여 [MASK]토큰을 생성한 뒤 원래의 단어를 예측하도록 학습하는 것이고, 또 다른 하나는 두 개의 문장을 입력으로 주어 각 문장의 연관성을 학습하는 것이다.

이러한 BERT모델을 기반으로 암호화된 페이로드 전체를 사전학습 과정과 파인튜닝 과정을 통해 패킷 5개와 패킷 1개를 입력 데이터로 각각 정확도 0.9729, 0.9890을 달성[15]했다. 오랜 사전학습 과정과 패킷을 통째로 입력데이터로 사용하는 만큼, 연산에 필요한 자원과 시간이 막대했을 문제점이 예상되었다.

2.2. 연구배경

2.2.1 데이터 경량화

연대기적 관점에서 우선적으로 눈여겨 볼만한 연구 중에 하나는 직관적으로 처음 몇 개의 패킷이 애플리케이션의 연결단계에서의 중요한 정보를 담고 있기 때문에 트래픽분류의 중요한 점으로 짚어낸 연구[4]이다. 각 TCP 플로우의 처음 5개 패킷을 사용하여 여러 애플리케이션에 대해 전체 플로우의 80% 이상의 정확도를 보여주었다.

다만 예외적으로 POP3 애플리케이션의 경우는 클러스터를 구성하지 않는 점 때문에 정확도가 떨어지지만, 이 연구의 결과에 따라 트래픽의 플로우를 조기에 감지하는데 좋은 교훈을 배우게 된 점은 이전부터도, 지금도[15] 주목받고 있다. 따라서 우리는 여기에 기반하여 플로우의 처음 5개 패킷으로 이루어진 플로우를 입력 데이터로 활용했다.

또한 트래픽 분류모델에서 새롭지만 반드시 필요한 실시간 트래픽분류를 하기 위한 시도가 등장했다. 여러 개의 패킷이 연결된 플로우 수준의 데이터를 입력으로 받지 않고, 단위 패킷 수준만을 데이터로 입력받는 모델을 제안한 연구가 제시되었다.

[11]에서는 트래픽 분류를 위해 애플리케이션 계층에서 헤더 정보 외에도 페이로드를 일부만 사용해도 페이로드에 잠재적으로 민감한 사용자 활동 정보가 포함되어 있기 때문에 정보보호의 문제를 제기하면서 페이로드 첫 40, 50, 60bytes 중에 분류 성능 및 보안적 관점에서 최적값을 찾고자 하였다. 실험을 통해 40bytes에 대비해서 50bytes는 성능향상이 크게 있지만, 60bytes는 성능적 향상이 없음을 제시했다. 성능과 보안적 측면에서 헤더 정보 외에 추가적으로 페이로드 50bytes만을 추가적으로 입력 데이터의 최적값으로 선택하기도 했다.

2.2.2 경량화된 모델

BERT는 뛰어난 성능과 간단한 파인튜닝 기법에도 불구하고 거대한 모델크기(파라미터 개수), 느린 추론 속도, 복잡하고 비용이 많이 드는 사전학습 과정으로 인해 사용이 제한되어 모델을 경량화하고 추론 속도를 높이고자 하는 많은 연구가 있었다.

많은 head들을 제거하더라도 성능에 영향이 없다는 것을 밝혀낸 연구[16]도 있었다.
자연어처리 분야에서 Transformer를 기반으로 하는 모델에서 사용되는 multi-head attention은 각각의 head는 입력의 각기 다른 부분에 집중하도록 한다. 결과적으로 단순히 가중합을 사용하는 것보다 정교한 함수를 사용해 정보를 가공할 수 있다는 것에 대한 반론을 제시한 것이다.

성능이 좋고 크기가 큰 교사 모델의 결과를 작은 학생모델에 가르치는 접근방법인 Knowledge Distillation(KD)[17]이 등장하면서 BERT 경량화의 흐름이 바뀌었다. 각 로짓에 대해 temperature를 조절함으로써 일반적인 소프트맥스에 비해 정답이 아닌 라벨에도 작은 확률값을 부여함으로써 테스트 데이터에서 일반화 성능을 높이는 방법이었다.

이러한 KD연구에 기반하여 BERT의 크기를 40%까지 줄이면서도, 성능은 97%를 유지하고, 추론속도는 60% 정도 향상된 실효성을 가지는 것이 DistilBERT 모델 연구에 제시되어 있다.
따라서 본 논문에서는 트래픽 데이터를 경량화하고, 이렇게 경량화된 데이터를 NLP분야에서 우수한 성능을 가지는 BERT 모델이 경량화된 DistilBERT a 모델에 적용하였다. NLP분야의 모델을 네트워크 트래픽 분류에 맞춰 활용하고자 한 것이다.

Ⅲ. 데이터 및 모델 구성

3.1. 데이터셋 선정

인터넷에 공개되어 다른 논문들에서도 공통적으로 활용되고 있는 데이터셋인 VPN-nonVPN dataset (ISCXVPN2016)[17]을 선택했다.

그 이유 중 하나는, 실제 사용되는 데이터를 수집해 Ground-truth 특성을 가지고 있다는 점이다. 또 다른 이유는 이전에 이루어진 많은 논문들이 제시한 실험결과와 비교가 가능하여 본 연구가 실질적인 의미에서 가치 있을 것으로 판단했기 때문이다.

공개데이터 분포 비율을 유지한 채 샘플데이터로 6개 클래스의 데이터를 구성한 것인데, 학습과 분류에 소요되는 자원과 시간을 고려한 선택이었다. 아래의 표 1은 트래픽 분류범위 안에 포함된 어플리케이션의 종류를 나타낸 것이다.

표 1. 트래픽 분류범위 내 애플리케이션의 구성
Table 1. Composition of applications within traffic catagories

Traffic Categories	Applications
Email	SMTP/S, POP3/SSL and IMAP/SSL
Chat	AIM, Facebook, Gmail, Hangouts, ICQ and Skype
Streaming	Netflix, Spotify, Vimeo and Youtube
File Transfer	FTPS, SCP, SFTP and Skype
VoIP	Facebook, Skype, Hangouts(voice and video calls) and VoipBuster
P2P	Bittorrent

다음 그림 1은 6개 클래스의 데이터 27,811개로 구성되어 있으며, 그에 대한 수량적 비율 분포도를 파이차트로 나타낸 것이다. 각 클래스가 균형적이지 않고 사용자의 사용성에 따라 각 클래스별 데이터가 다른 것이 특징적이다.

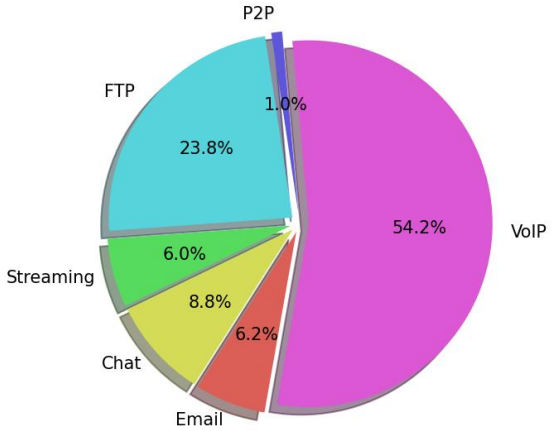


그림 1. 6개 클래스로 구성된 27,811개의 데이터 분포비율
Fig. 1. Distribution ratio of 27,811 data composed of 6 classes

3.2. 데이터 전처리 및 리프젠테이션

[18], [19]에서 모델에 입력되기 전에 로컬 최소값을 상위 수준의 추상화인 내부 분산표현을 생성함으로써 최적화에 도움이 된다는 연구결과를 제시하였다. 따라서 로컬 최소값을 의미하는 데이터에 대한 전처리 및 리프리젠테이션은 딥러닝에서 중요한 역할을 한다.

그래서 우리는 [20]에서 사용하는 방법을 준용하여 ethernet header와 IP header에 대한 처리, DNS 또는 페이로드가 없는 패킷에 대한 처리, 패딩 및 정규화를 적용했다. 전처리 알고리즘은 아래 그림 2의 알고리즘 1과 같다.

Algorithm 1. Proprocessing

```
Input : PATH = continuous bi-directional packets, MAX_PACKET = 100, MAX_ROW = 5
Output : packets_row = preprocessed data from pcap file
1 : packets_row = [ ]
2 : for i, packet in enumerate(read_pcap(PATH))
3 :   if i >= 0 and i < MAX_ROW
4 :     packet = discard_packet(packet)        # no payload TCP or DNS
5 :     packet = remove_ethernet_header(packet)
6 :     if udp == categorize_packet(packet)        # distinguish protocol
7 :         packet = pad_udp(packet)        # with zeros to length of 20bytes
8 :     packet = mask_ip(packet)        # src, dst : 0.0.0.0
9 :     packet = normalization(packet)
10 :                # min-max scale with values between 0 and 1 for each element
11 :                # fix the packet into MAX_PACKET bytes using zero_padding or cutting
12 : return packets_row        # for saving the proprocessed data
```

그림 2. 전처리 알고리즘
Fig. 2. Preprocessing algorithm

모델에 입력하기 위한 단위로 패킷 1개는 앞 100bytes로 구성되어 있으며, 양방향 플로우(세션)인 패킷 5개의 100bytes가 연결된 구성이다.

데이터셋을 train과 test로 분할시 비율은 8:2로 처리했다.

3.3. 실험전 셋팅

앞에서 제시된 데이터의 클래스 불균형은 모델의 학습과정에서 오버피팅을 유발할 수 있기 때문에 학습 전에 사전 처리가 필요하다.

[21]에는 클래스 불균형 문제가 언급되어 있는데, 주 표본의 분류에만 집중하면서 소수 표본을 무시하거나 오분류하는 실수가 이루어질 수 있어서 균형을 맞추는 3가지 범주의 방법이 제시됐다. 우리는 그중에 클래스 가중치를 사용하여 학습 과정에서 서로 다른 클래스 분포의 균형을 맞추어 학습하는 방법을 선택했다.

계산식은 아래 (1)과 같으며, 이를 적용해 얻은 상대적인 클래스 웨이트 값은 그림 3과 같다.

$$W = 1 - \frac{Q_i}{\sum_i Q_i} \quad (1)$$

W는 클래스 웨이트, Q_i는 i번째 클래스의 데이터 수량을 의미한다.

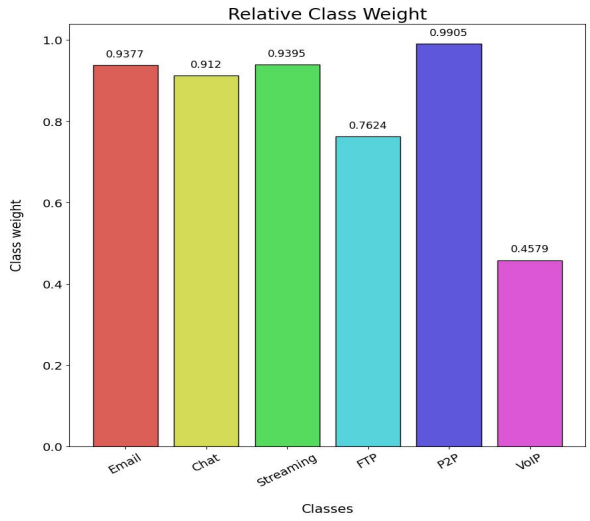


그림 3 상대적 클래스 웨이트

Fig. 3. Relative Class Weight

모델의 학습과정에 클래스 웨이트가 미치는 영향을 비교한 실험은 '4.3 추가적인 실험과 분석' 부분에 제시되어 있다.

3.4. 모델 구성

전처리과정을 통해 리프리젠테이션된 패킷이 토크나이저를 거쳐 DistilBERT를 통해 학습 및 분류되는 과정을 아래 그림 4와 같이 단계적으로 나타내었다.

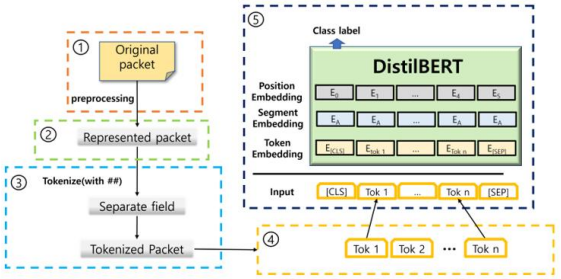


그림 4. 전체적인 모델 구성

Fig. 4. Overall model configuration

순차적인 진행 과정은 다음과 같다.

1) 전처리 과정을 통해 로컬 최소값을 상위 수준의 추상화인 내부 분산표현으로 생성함.
2) 전처리시 오버피팅을 없애고 모델의 학습효과를 높이도록 리프리젠테이션도 동시수행 함.
3) bytes 단위로 구분된 각 필드의 값을 토크나이징하여 vocabulary화 함.
4) 입력의 맨 앞, 뒤에 스페셜 토큰을 추가하여 최종 토큰들로 최종 변환함.
5) 최종 입력 데이터를 사용해 DistilBERT 모델이 학습하고 추론함.

표 2. 모델 내부의 순차적인 데이터 구조

Table 2. Sequential data structure inside the model

Step	Example Data	Size of Data
①	A = [d_1, … ,d_{100}]	100 bytes
②	A′ = [d'_1, … ,d'_{100}]	
③	A″ = [d''_1, … ,d''_{100}]	
④		
⑤	A‴ = [CLS, d''_1, … , d''_{98}, SEP]	100 bytes = 98 bytes (represented data) + 2 bytes (special token)

앞의 표2는 첫 패킷의 앞 100bytes의 최초의 입력 데이터가 전처리 과정에서부터 모델에 입력되기까지의 데이터가 가지는 구조와 크기를 예시적으로 나타낸 것이다.

모델의 학습이 효과적으로 이루어지도록 선택한 옵티마이저는 RAdam이다. RAdam[21]은 Adam[22]이 가지고 있는 bad local optima problem을 해결하고자 Adam의 adaptive learning rate term의 분산을 바로잡아, 적은 학습 단계에서도 학습 안정성을 높인 옵티마이저이다. RAdam을 사용함으로써 학습 단계(에포크)에서 빠르고 안정성 있게 학습하는 성능은 '4장 실험 결과 및 분석' 부분에 제시되었다.

Ⅳ. 실험 결과 및 분석

4.1. 실험 환경

모델은 CUDA 11.3 기반의 TensorFlow 2.9.2, Python 3.7.14 및 Pytorch 1.12.1를 기반으로 구현되었다. 하드웨어는 4Core Intel(R) Xeon(R) CPU @ 2.30GHz의 CPU와 NVIDIA Tesla P-100(16GB 메모리)의 GPU가 장착된 서버로 구성되었다.

4.2. 실험결과 및 분석

모델을 평가하기 위한 요소로 사용된 지표들은 아래 (2), (3), (4) 및 (5)입니다.

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \quad (2)$$

$$\mathrm{Precision} = \frac{TP}{TP + FP} \quad (3)$$

$$Recall = \frac{TP}{TP + FN} \quad (4)$$

$$F1 - score = 2 \times \frac{\mathrm{Precision} \times Recall}{\mathrm{Precision} + Recall} \quad (5)$$

위에서부터 순서대로 식 (2)에 나오는 용어의 의미이다. True Positive(TP)는 실제 True인 정답을 True라고 예측(정답)하는 것이다. False Positive(FP)는 실제 False인 정답을 True라고 예측(오답)하는 것이다. False Negative(FN)는 실제 True인 정답을 False라고 예측(오답)하는 것이다. 마지막으로 True Negative(TN)는 실제 False인 정답을 False라고 예측(정답)하는 경우이다.

다음 그림 5, 6은 우리 모델의 학습을 통한 예측 성능을 나타낸 컨퓨전 메트릭스와 각 평가요소이다.

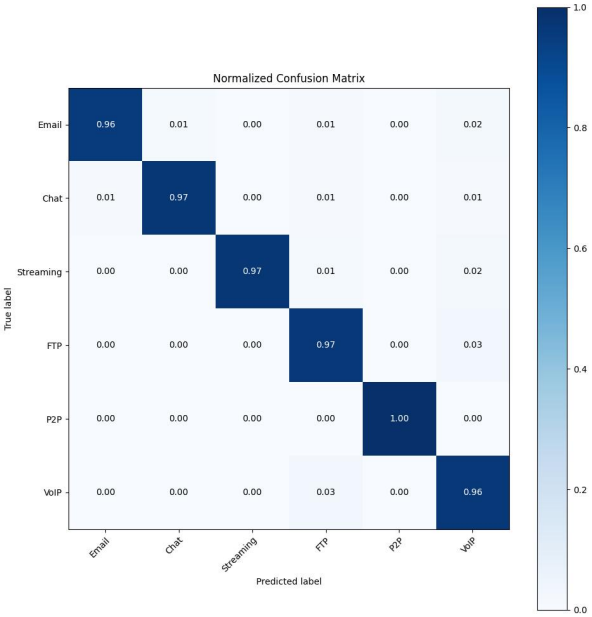


그림 5. 컨퓨전 메트릭스
Fig. 5. Confusion Matrix

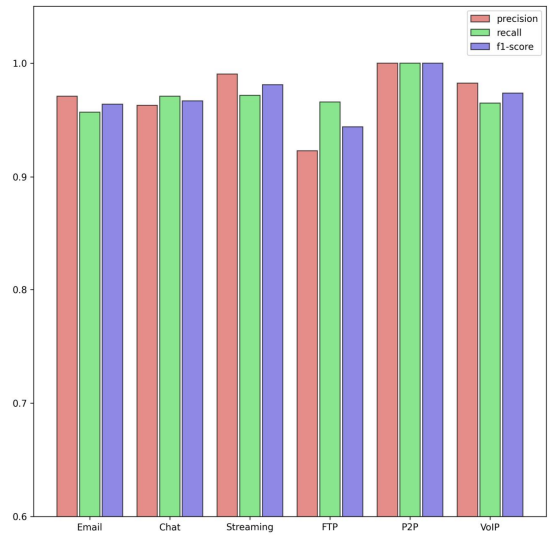


그림 6. 정밀도, 재현율, F1-score
Fig. 6. Precision, Recall, F1-score

그림 5에서 보면, 6개 클래스의 트래픽을 분류한 결과 모두 0.96 ~ 1.0 값을 나타내어 Accuracy가 고르게 높은 모습이 보인다. 그림 6에서 보면, FTP 트래픽을 제외하고는 5개 클래스 모두 Precision, Recall, F-1-score 모두 고르게 0.95 이상을 상회하는 모습을 볼 수 있다.

아래의 표3는 동일한 데이터셋인 VPN-nonVPN dataset (ISCXVPN2016)을 사용하는 다른 모델들과 정확도와 F1-score 값을 통해 성능을 비교한 것이다. [8]에서는 1500bytes 길이의 패킷 10개로 구성된 데이터를 LSTM과 Attention Mechanism(AM)이 조합된 모델에 적용해 얻은 성능이다. 더 긴 패킷 길이와 더 많은 패킷 수를 사용했음에도 우리의 성능에 못 미친다.
[11]에서는 90bytes(헤더 40bytes + 페이로드 50bytes)길이로 구성된 데이터를 self-attention이 적용된 모델에 적용해 얻은 성능이다. 조금 짧은 패킷 길이를 적용했지만 우리의 성능에 크게 못 미치는 것을 확인할 수 있다.

표 3. 동일한 데이터셋을 활용한 다른 연구들과의 성능비교 (정확도, F1-score)
Table 3. Performance comparison with other studies using the same dataset

	Unit	# of classes	Accuracy / F1-score
[8]	session	12	0.9120 / -
		6	0.9480 / -
[11]	packet	6	0.9033 / 0.8560
Proposed	session	6	**0.9703 / 0.9706**
	packet		**0.9707 / 0.9731**

표 4에서는 모델의 총 파라미터 개수 측면에서 우리가 선택한 DistilBERT가 일반적인 BERT모델이 가지는 파라미터보다 훨씬 적은 수의 파라미터를 사용하는 것을 확인할 수 있다. 경량화된 데이터로 DistilBERT를 활용해 파인튜닝하는 학습과정 후, 테스트 시간은 0.0099 second/packet임에 따라 패킷 단위 분류에 적은 시간이 소요됨을 확인했다.

표 4. 모델의 총 파라미터 갯수
Table 4. Total number of parameters in model

	BERT	DistilBERT
# of parameter	177,853,440	**66,965,007**

4.3. 추가적인 실험과 분석

그리고 멀티클래스 데이터가 클래스 불균형을 이룰 때 모델의 성능을 비교하는 지표는 Accuracy가 아닌 F1-score임에 따라, 추가적으로 표 5의 성능지표를 확인했는데, 모델의 뛰어난 학습성능으로 인해 결과적 수치로는 큰 차이가 없었다.
그러나 로스 함수에 클래스 웨이트를 적용하는 여부에 따른 학습 곡선을 아래의 그림 7, 8에서 비교해보면, 클래스 웨이트가 적용된 모델의 학습 곡선이 비교적 안정적인 모습임을 확인할 수 있다.

표 5. 로스함수에 클래스 웨이트 적용여부에 따른 성능비교
Table 5. Performance comparison according to whether or not class weight is applied to the loss function

	Class Weight	Non Class Weight
Accuracy / F1-score	0.9705 / 0.9746	0.9703 / 0.9742

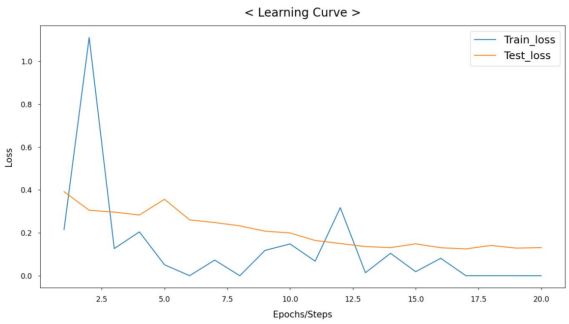


그림 7. 클래스 웨이트를 적용한 모델의 학습 곡선
Fig. 7. Learning curve of a model with class weights

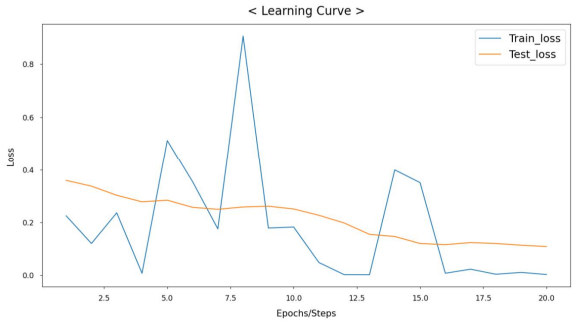


그림 8. 클래스 웨이트를 미적용한 모델의 학습 곡선
Fig. 8. Learning curve of a model without class weights

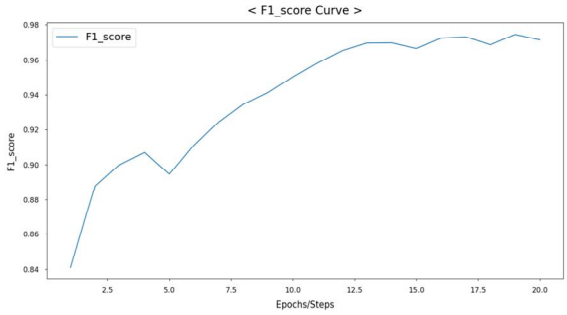


그림 9. 클래스 웨이트를 적용한 모델의 F1-score 곡선
Fig. 9. F1-score curve of the model with class weights applied

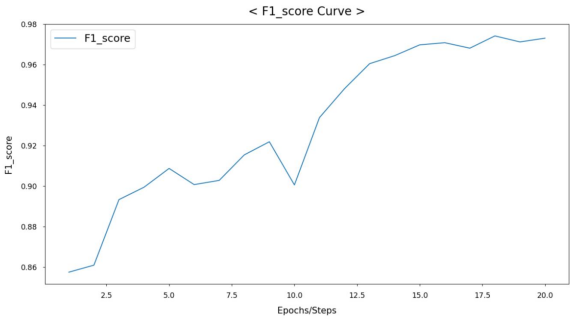


그림 10. 클래스 웨이트를 미적용한 모델의 F1-score 곡선
Fig. 10. F1-score curve of the model without class weights

동일한 관점에서 로스 함수에 클래스 웨이트를 적용하는 여부에 따라, F1-score 값의 변화 곡선을 위의 그림 9, 10을 통해 비교해 보았다. 클래스 웨이트가 적용된 모델의 F1-score 값이 보다 빠르고 안정적이게 증가함을 확인할 수 있다.

또한 실험 간에 확인된 사실은 27,811개의 데이터 중 80%로 구성된 22,248개의 학습데이터로 1에포크의 training에 소요된 시간은 4분 45초였고, 사전학습이 불필요했다는 점을 감안해 보면 굉장히 짧은 학습 시간임을 알 수 있다.

V. 결론 및 향후 연구방향

5.1. 결론

트래픽분류는 컴퓨터 네트워크 영역에서 서비스 관리 및 보안 등의 분야에서 그 역할이 점점 더 중요해지고 있다. 초기에는 포트넘버, DPI, 통계정보 등을 활용해 트래픽 분류가 가능했다. 그러나 정보보호 측면에서 트래픽의 페이로드가 암호화되면서 앞서의 기법들로는 좋은 성능을 달성할 수 없었다. 그에 따라 머신러닝기법을 추가 활용하면서, 여러 가지 조합에 따라 선택된 변수(통계정보, 시그니처 등)들과 함께 트래픽분류는 한걸음 나아갔다. 이후 AE, CNN, RNN 등의 다양한 딥러닝 모델들을 활용한 트래픽 분류도 계속 발전되었다.

이후 RNN의 단점을 개선한 Transformer 모델이 self-attention mechanism이 제시되었고, 매우 우수한 성능을 보여줌에 따라 딥러닝은 또 한 걸음 나아갔다. 따라서 NLP분야에서 우수한 성능을 보이는 Transformer를 기반으로 다양한 분야에서 딥러닝이 확장되고 있다.

하지만 딥러닝에 계속 발전하면서 장점이자 단점으로, 변수 선택은 그 역할이 줄어들고 수 많은 변수들로 구성된 데이터를 활용해도 트래픽 분류가 문제없어짐에 따라, 딥러닝 모델과 데이터가 점차 커졌다. 이에 따라 큰 모델에는 자원과 시간이 많이 소모됨에 따라 부담스럽고 성가시게 되었다.

이렇게 부담스럽고 성가신 모델과 데이터를 경량화하여 효율적인 네트워크 트래픽 분류가 필요했다. 본 연구에서는 KD 연구결과에 따라 제시되었던 모델 경량화 기법이 적용된 DistilBERT 모델에 경량화한 데이터를 적용하는 것이 중요하다고 판단했다. 그리고 모델의 선택에 있어서 동시적으로 고려된 점으로, 트래픽 데이터가 시계열적 연속성을 가짐에 따라 NLP모델을 트래픽분류에 적용하는 것이 가치 있을 것이라는 점이다.

이에 따른 실험결과로 플로우의 첫 번째 패킷 1개의 앞 100bytes 입력과 그 패킷 5개를 연결한 입력으로 정확도 / F1-score가 각각 0.9707 / 0.9731과 0.9703 / 0.9706를 보여, 현재까지 제시된 다른 연구들에 대비해 매우 우수한 성능임을 확인했다.

5.2. 향후 연구방향

이후에 암호화된 네트워크 트래픽 분류분야에 딥러닝을 적용하는 측면에서, 우리의 연구 방향은 크게 다음 3가지 정도로 판단하고 있다.

첫째, 패킷과 플로우를 NLP관점에서 처리하는 방법과 이미지관점에서 처리하는 방법을 적용해 보고 장·단점을 비교해 보는 것이다. 두 번째, 전처리과정에서 암호화된 페이로드가 가지는 함축된 정보를 전달할 수 있도록 요약적 페이로드를 적용해 보는 것이다. 마지막으로 입력데이터인 패킷과 플로우를 모델에서 보다 효율적으로 인식할 수 있도록 모델의 메커니즘을 개선해 보는 것이다.

References

[1] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, “Characterization of Encrypted and VPN Traffic using Time-related Features:,” in Proceedings of the 2nd International Conference on Information Systems Security and Privacy, Rome, Italy, 2016, pp. 407 –414. doi: 10.5220/0005740704070414.

[2] A. W. Moore and K. Papagiannaki, “Toward the Accurate Identification of Network Applications,” in Passive and Active Network Measurement, vol. 3431, C. Dovrolis, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 41–54. doi: 10.1007/978-3-540-31966-5_4.

[3] T. T. T. Nguyen and G. Armitage, “A survey of techniques for internet traffic classification using machine learning,” IEEE Communications Surveys Tutorials, vol. 10, no. 4, pp. 56–76, 2008, doi: 10.1109/SURV.2008.080406.

[4] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian, “Traffic classification on the fly,” SIGCOMM Comput. Commun. Rev., vol. 36, no. 2, pp. 23–26, Apr. 2006, doi: 10.1145/1129582.1129589.

[5] J. Erman, A. Mahanti, M. Arlitt, and C. Williamson, “Identifying and discriminating between web and peer-to-peer traffic in the network core,” in Proceedings of the 16th international conference on World Wide Web - WWW ’07, Banff, Alberta, Canada, 2007, p. 883. doi: 10.1145/1242572.1242692.

[6] J. Höchst, L. Baumgärtner, M. Hollick, and B. Freisleben, “Unsupervised Traffic Flow Classification Using a Neural Autoencoder,” in 2017 IEEE 42nd Conference on Local Computer Networks (LCN), Oct. 2017, pp. 523–526. doi: 10.1109/LCN.2017.57.

[7] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, “Network Traffic Classifier With Convolutional and Recurrent Neural Networks for Internet of Things,” IEEE Access, vol. 5, pp. 18042–18050, 2017, doi: 10.1109/ACCESS.2017.2747560.

[8] H. Yao, C. Liu, P. Zhang, S. Wu, C. Jiang, and S. Yu, “Identification of Encrypted Traffic Through Attention Mechanism Based Long Short Term Memory,” IEEE Transactions on Big Data, vol. 8, no. 1, pp. 241–252, Feb. 2022, doi: 10.1109/TBDATA.2019.2940675.

[9] A. Vaswani et al., “Attention is all you need,” in Proceedings of the 31st International Conference on Neural Information Processing Systems, Red Hook, NY, USA, Dec. 2017, pp. 6000–6010.

[10] HaoLi, “Traffic classification algorithm using CNN and multi-head attention mechanism for representation learning,” J. Phys.: Conf. Ser., vol. 2258, no. 1, p. 012001, Apr. 2022, doi: 10.1088/1742-6596/2258/1/012001.

[11] G. Xie, Q. Li, and Y. Jiang, “Self-attentive deep learning method for online traffic classification and its interpretability,” Computer Networks, vol. 196, p. 108267, Sep. 2021, doi: 10.1016/j.comnet.2021.108267.

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” arXiv, May 24, 2019. doi: 10.48550/arXiv.1810.04805.

[13] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, “DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter.” arXiv, Feb. 29, 2020. Accessed: Sep. 15, 2022. [Online]. Available: http://arxiv.org/abs/1910.01108

[14] S. Rezaei and X. Liu, “Multitask Learning for Network Traffic Classification,” in 2020 29th International Conference on Computer Communications and Networks (ICCCN), Aug. 2020, pp. 1–9. doi: 10.1109/ICCCN49398.2020.9209652.

[15] X. Lin, G. Xiong, G. Gou, Z. Li, J. Shi, and J. Yu, “ET-BERT: A Contextualized Datagram Representation with Pre-training Transformers for Encrypted Traffic Classification,” Feb. 19, 2022. Accessed: Apr. 02, 2022. [Online]. Available: http://arxiv.org/abs/2202.06335

[16] P. Michel, O. Levy, and G. Neubig, “Are

Sixteen Heads Really Better than One?" arXiv, Nov. 04, 2019. Accessed: Dec. 02, 2022. [Online].Available:http://arxiv.org/abs/1905.10650

[17] G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network." arXiv, Mar. 09, 2015. Accessed: Sep. 16, 2022. [Online].Available:http://arxiv.org/abs/1503.02531

[18] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy Layer-Wise Training of Deep Networks," in Advances in Neural Information Processing Systems, 2006, vol. 19. Accessed: Dec. 02, 2022. [Online]. Available: https://proceedings.neurips.cc/paper/2006/hash/5da713a690c067105aeb2fae32403405-Abstract.html

[19] G. Zhong, L.-N. Wang, and J. Dong, "An Overview on Data Representation Learning: From Traditional Feature Learning to Recent Deep Learning." arXiv, Nov. 24, 2016. Accessed: Dec. 02, 2022. [Online]. Available: http://arxiv.org/abs/1611.08331

[20] M. Lotfollahi, R. S. H. Zade, M. J. Siavoshani, and M. Saberian, "Deep Packet: A Novel Approach For Encrypted Traffic Classification Using Deep Learning," arXiv:1709.02656 [cs], Jul. 2018, Accessed: Mar. 16, 2022. [Online]. Available: http://arxiv.org/abs/1709.02656

[21] L. Liu et al., "On the Variance of the Adaptive Learning Rate and Beyond." arXiv, Oct. 25, 2021. doi: 10.48550/arXiv.1908.03265.

[22] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization." arXiv, Jan. 29, 2017. doi: 10.48550/arXiv.1412.6980.

신 창 의 (Chang-Yui Shin)

2003년 : 육군사관학교 운영분석학과 학사

2007년 : 고려대학교 전자컴퓨터 공학과 석사

2022년~현재 : 고려대학교 컴퓨터정보학과 박사과정

<관심분야> 네트워크 관리 및 보안, 트래픽 모니터링 분석

박 지 태 (Jee-Tae Park)

2017년 : 고려대학교 컴퓨터정보학과 학사

2017년~현재 : 고려대학교 컴퓨터정보학과 석박사통합과정

<관심분야> 네트워크 관리 및 보안, 트래픽 모니터링 분석

백 의 준 (Ui-Jun Baek)

2018년 : 고려대학교 컴퓨터정보학과 학사

2018년~현재 : 고려대학교 컴퓨터정보학과 석박사통합과정

<관심분야> 네트워크 관리 및 보안, 트래픽 모니터링 및 분석

최 정 우 (Jung-woo Choi)

2018년 : 고려대학교 컴퓨터정보학과 학사

2022년~현재 : 고려대학교 컴퓨터정보학과 석사과정

<관심분야> 네트워크 관리 및 보안, 트래픽 모니터링 및 분석

김 명 섭 (Myung-Sup Kim)

1998년 : 포항공과대학교 전자계산 학과 학사

2000년 : 포항공과대학교 전자계산 학과 석사

2004년 : 포항공과대학교 전자계산 학과 박사

2006년 : Dept. of ECS, Univof Toronto Canada

2006년~현재 : 고려대학교 컴퓨터정보학과 교수

<관심분야> 네트워크 관리 및 보안, 트래픽 모니터링 및 분석, 멀티미디어 네트워크

KNOM Review '22-02 Vol.25 No.02
https://doi.org/10.22670/knom.2022.25.2.51

네트워크 트래픽 분석을 통한 규칙 기반 사용자 행위 탐지 시스템 설계

박 지 태◆, 백 의 준*, 신 창 의**, 최 정 우*, 김 명 섭°

A Design of Rule-based User Action Detection System for Network Traffic Analysis

Jee-Tae Park◆, Ui-Jun Baek*, Chang-Yui Shin**, Jeong-Woo Choi*, Myung-Sup Kim°

요 약

최근 네트워크 기술과 환경의 성장에 따라 다양한 어플리케이션이 발생하고 있으며, 네트워크 트래픽 양도 급격하게 증가하고 있다. 이러한 추세에 따라 네트워크 내 원활한 서비스 제공과 안전한 네트워크 보안을 위해서는 효율적인 네트워크 관리 방법이 필요하며, 이를 위해 오래전부터 다양한 연구가 수행되어 왔다. 여러 연구들 중 사용자 행위 탐지 연구는 네트워크 내 사용자들의 어플리케이션 사용 행위에 대한 탐지 및 모니터링을 수행하며, 네트워크 관리, 보안 및 지출 관리 등의 여러 분야에 도움을 준다. 본 논문에서는 정확한 사용자 행위 탐지를 목표로 규칙 기반의 사용자 행위 탐지 시스템을 제안한다. 제안하는 방법의 타당성을 검증하기 위해 실제 어플리케이션 트래픽을 수집하고, 본 연구의 선행 연구와 성능 비교 실험을 수행한다.

Key Words : Network Traffic Classification, User Action Detection, SaaS Application

ABSTRACT

Recently, various applications are occurring according to the growth of network technology and environment, and the amount of network traffic is rapidly increasing. In accordance with this trend, an efficient network management is required for smooth service provision and safe network security in the network, and various researches have been conducted for a long time for this purpose. Among several researches, user action detection research detects and monitors application usage behavior of users within the network, and helps in various fields such as network management, security, and expenditure management. In this paper, we propose a rule-based user action detection system with the goal of accurate user action detection. In order to verify the validity of the proposed method, we collect real application traffic and conducted performance comparison experiments with our previous research.

※ 이 논문은 2020년도 산업통상자원부 및 한국산업기술평가관리원(KEIT) 연구비 지원에 의한 연구이며 (No. 20008902, IT비용 최소화를 위한 5채널 탐지기술 기반 SaaS SW Management Platform(SMP) 개발), 2021년도 교육부의 재원으로 한국연구재단의 지원을 받아 수행된 지자체-대학 협력기반 지역혁신 사업의 결과로 수행되었음 (2021RIS-004)
◆ First Author : Department of Computer and Information Science, Korea University, pjj5846@korea.ac.kr
° Corresponding Author : Department of Computer and Information Science, Korea University, tmskim@korea.ac.kr
* Department of Computer and Information Science, Korea University, {pb1069, choigoya97}@korea.ac.kr
** Defense Agency for Technology and Quality, superego99@dtaq.re.kr
논문번호 : KNOM2022-02-10, Received December 2, 2022; Revised December 10, 2022; Accepted December 16, 2022

Ⅰ. 서 론

오늘날에는 네트워크 환경의 증대와 기술의 발전으로 다양한 어플리케이션이 등장하고 있으며, 모바일, 클라우드 기반 등의 새로운 형태의 어플리케이션도 나타나고 있다[1-3]. 네트워크 서비스를 원활하게 제공하기 위해서는 효율적인 네트워크 관리 방안이 필요하며, 오래전부터 네트워크 관리를 위한 다양한 연구가 수행되어 왔다. 그 중 사용자 행위 탐지 연구는 네트워크 내 보안 및 관리 측면에서 중요한 역할을 한다. 관리자는 네트워크 트래픽을 입력으로 대상 어플리케이션에 대한 실제 사용 행위를 모니터링하며, 이를 통해 네트워크 관리에 유용한 여러 가지 정보를 얻을 수 있다. 또한, 해커들은 경제적, 정치적 등의 다양한 목적으로 공격 대상에게 악성 행위를 수행하며, 이러한 악성 행위는 점차적으로 고도화 되고 있다. 특히 IDS와 같이 네트워크 보안 관제 시스템에 탐지되지 않기 위해 사전에 공격 대상의 취약점을 분석하며, 이를 위해 수행되는 악성 행위가 정상 행위인 것처럼 위장한다[2,3]. 네트워크 보안 관제 시스템은 네트워크 내 사용자 행위에 대한 지속적인 모니터링을 통해 정상 행위로 위장된 악성 행위를 신속하고 정확하게 탐지해야 한다.

사용자 행위 탐지에 대한 연구는 각 연구 별로 목적에 따라 다르게 수행되어 왔으며, 대상 어플리케이션의 특성에 따라 사용자 행위 정의도 다르게 나타난다. 사용자 행위 정의는 행위 탐지 이전에 어떤 행위를 탐지 할 것인지 정의하는 것으로 다양하게 정의될 수 있다. 예를 들어 WeChat과 같은 메신저 형태 어플리케이션의 경우, 채팅방에 들어가기, 채팅 시작하기, 파일 전송 등과 같은 행위가 정의될 수 있으며, Twitter와 같은 SNS 형태 어플리케이션의 경우 게시물 올리기, 게시물 공유 등과 같은 행위가 정의 될 수 있다.

본 논문에서는 클라우드 서비스를 기반으로 하는 SaaS(Software as a Service) 어플리케이션을 대상으로 사용자 행위 탐지 연구를 수행한다. 클라우드 서비스는 네트워크가 연결되어 있는 환경에서 가상화된 컴퓨터 리소스를 여러 가지 서비스 형태로 제공하며, 그 중 SaaS는 리소스를 소프트웨어 형태로 제공하는 클라우드 서비스이다. 과거에는 사용자가 대상 어플리케이션을 1회성 구매 후 1회 설치를 통해 사용하였지만, SaaS의 경우 네트워크가 연결되어 있는 환경에서 라이선스 구독 후, 구독한 기간에 따라 사용 할 수 있다. 사용자 입장에서 번거로운 설치 과정 없이 바로 사용 가능하기 때문에 편리하며, 적절한 라이선스, 기간, 인원을 선택 할 경우에 기존의 1회성 구매에 비해 비용을 절감 시킬 수 있다는 장점이 있다. 특히 비용 절감은 회사, 학교와 같은 대규모의 네트워크 환경에서 더욱 크게 적용하기 때문에, 최근에는 많은 단체에서 SaaS를 사용하는 추세이다. 하지만 실제 사용하는 인원에 비해 더 많은 허용 인원을 가진 라이선스 혹은 필요하지 않은 기능이 포함된 라이선스를 구독할 경우 불필요한 지출이 발생할 수 있다. 따라서 SaaS 어플리케이션을 사용 할 때, 불필요한 지출을 줄이고 적절한 비용 관리를 위해 SaaS 어플리케이션에 대한 사용자 행위 탐지를 기반으로 하는 사용자 행위 모니터링이 필요하다.

본 논문에서는 클라우드 서비스를 기반으로 하는 SaaS 어플리케이션을 대상으로 효율적인 네트워크 관리와 불필요한 지출을 줄이는 것을 목표로 사용자 행위 탐지 시스템을 설계 한다. 먼저 대상 어플리케이션에 대한 4 가지의 사용자 행위를 사전에 정의하고, 대상 어플리케이션에 대한 트래픽 분석을 수행한다. 분석 결과를 바탕으로 사전에 정의한 사용자 행위를 탐지하기 위해 규칙 기반 행위 탐지 시스템을 제안한다.

본 논문은 1장 서론에 이어 2장에서는 관련 연구를 설명하고, 3장에서 SaaS 어플리케이션에 대한 행위 장의 및 제안하는 시스템의 전반적인 구조에 대해 설명한다. 4장에서는 제안하는 시스템을 검증하기 위해 수행된 실험에 대해 기술하며, 5장에서는 결론 및 향후 연구에 대해 설명하고 마친다.

Ⅱ. 관련 연구

2.1. 네트워크 어플리케이션 트래픽 분류

네트워크 어플리케이션 트래픽 분류 연구는 오래전부터 연구가 진행되어 왔으며, 대표적으로 포트 기반, 페이로드 기반 분석 방법이 있다[1-3]. 포트 기반 분석 방법은 고정된 포트 정보를 활용하여 어플리케이션을 분류하는 방법으로, 동적 포트 사용으로 현재는 잘 사용하지 않는 방법이다. 페이로드 기반 분석 방법은 암호화 되어있지 않은 페이로드 정보를 활용하여 고정된 페이로드 스트링 값을 활용하여 어플리

케이션을 분류하는 방법이다. 하지만 암호화 트래픽 사용의 증가로 페이로드 값이 모두 암호화되어 나타나기 때문에 현재는 잘 사용하지 않는다[2-5].

이러한 문제점을 해결하기 위해 학습 기반 분석 방법이 가장 활발하게 연구되고 있다[4, 5]. 이 방법은 어플리케이션 트래픽의 여러 가지 정보를 활용하여 특징들을 추출하고, 추출된 특징을 머신러닝 및 딥러닝 알고리즘을 활용하여 학습한다. 이 때, 페이로드 정보는 암호화되어있다고 가정하기 때문에 주로 플로우의 통계 정보나 헤더 정보 등을 활용하여 특징을 추출한다. 이 방법은 트래픽의 여러 정보를 고려하여 특징을 추출할 수 있으며, 분류 정확도가 다른 방법론에 비해 높다는 장점이 있다[4, 5]. 하지만 학습 특징에 대한 높은 의존도와 어플리케이션 트래픽 분류 분야에서 라벨링된 학습 데이터를 쉽게 구하기 어려우며, 많은 시간과 비용이 든다는 문제점이 있다[5].

2.2. 사용자 행위 탐지 연구

사용자 행위 탐지 연구는 오래전부터 네트워크 보안, 관리를 목적으로 수행되어 왔다[6-10]. 사용자 행위를 탐지하기 위해서는 먼저 대상으로 하는 어플리케이션의 사용자 행위를 정의해야 한다. 사용자 행위 정의는 어플리케이션의 형태(Ex. 메신저, 눈, 파일 작업 등)에 따라서 다양하게 정의 될 수 있다 또한 같은 어플리케이션이라고 하더라도 수행한 연구 목적에 따라 다르게 정의된다[6, 7].

논문 [8]에서는 메신저 형태의 카카오톡(KakaoTalk)을 대상 어플리케이션으로 선정하였다. 카카오톡은 메신저 기반의 신속성과 경량화를 반영하여 TCP/IP 기반의 독자적인 프로토콜 LOCO를 사용한다. 논문 [8]에서는 카카오톡을 대상으로 11 가지의 사용자 행위를 정의하고, 정의된 행위를 Random Forest 알고리즘을 활용하여 99.7%의 정확도로 분류한다.

논문 [9]에서는 메신저 형태의 WeChat 을 대상 어플리케이션으로 선정하였다. WeChat은 카카오톡과 유사하게 메신저 기반의 신속성과 경량화를 반영하여 HTTP, TCP 기반의 독자적인 프로토콜 MMTLS를 사용한다. 9 가지의 사용자 행위를 정의하고, 정의된 행위를 6 가지의 학습 기반의 알고리즘을 활용하여 비교 실험을 수행한다. 여러 가지 알고리즘 중 Random Forest에서 96%의 F1-measure로 가장 좋은 결과를 도출한다.

논문 [10]에서는 SNS 형태의 어플리케이션으로 Instagram을 대상으로 9 가지의 사용자 행위를 정의하고, 정의된 행위를 학습 기반의 SVM 알고리즘을 활용하여 행위를 탐지한다.

본 연구의 선행 연구인 논문 [11]에서 Adobe Creative Cloud를 활용하여 페이로드 시그니처를 추출하고 탐지 실험을 수행한다. 하지만 논문에서는 단순하게 Adobe Creative Cloud에 대한 추출된 페이로드 시그니처를 제시하고, 실제로 수행한 실험에 대해서는 객관적인 자료를 제시하고 있지 않다. 또한 SNI 정보만을 가지고 행위를 탐지 할 경우, 특정 행위가 고정되지 않은 SNI 정보를 사용하는 경우에 탐지를 못하거나 잘못 탐지 할 수 있다는 문제점이 있다.

따라서 본 논문에서는 SaaS 어플리케이션에 대한 사용자 행위 탐지를 목표로, 가장 널리 사용되는 SaaS 어플리케이션 중 하나인 Adobe Creative Cloud를 대상 어플리케이션으로 선정하였다. 또한 선행 연구인 [11]의 시그니처 기반의 행위 탐지 방법과 성능 비교 실험을 수행하여 본 논문의 타당성을 검증한다.

Ⅲ. 본 론

3.1. 사용자 행위 정의

본 장에서는 대상 어플리케이션에 대한 사용자 행위를 정의한다. 이전에 언급한대로 사용자 행위는 연구 목적에 따라 여러 가지로 정의 될 수 있다 본 논문에서는 SaaS 어플리케이션의 불필요한 지출 줄이는 것을 목표로 연구를 수행하기 때문에 사용자의 실사용 기록과 관련된 4 가지 행위(Ex. 어플리케이션 시작, 로그인, 로그아웃 어플리케이션 종료)를 탐지 할 사용자 행위로 정의한다.

3.2. 규칙 기반 행위 탐지 시스템 설계

제안하는 규칙 기반 행위 탐지 시스템의 전체 구조는 크게 두 가지 모듈 (i.e. 규칙 생성 및 행위 탐지 모듈)로 구성되어 있으며, 그림 1에 나타나있다.

규칙 생성 모듈은 대상 어플리케이션에 대한

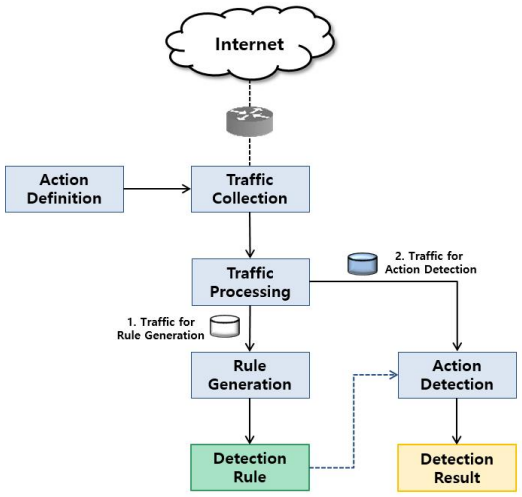


그림 1. 규칙 기반 행위 탐지 시스템 구조
Fig.1. A Structure of Rule based Action Detection System

분석 후 각 행위 별로 공통으로 발생하는 여러 가지 트래픽 정보를 활용하여 탐지 규칙을 생성하는 과정이며, 행위 탐지 모듈은 수집 된 트래픽과 생성된 탐지 규칙을 입력으로 실제 사용자 행위에 대한 탐지를 수행하는 과정이다.

먼저 Action Definition은 이전의 3.1장에서 설명한대로 대상 어플리케이션의 탐지 행위를 정의한다. 다음으로 Traffic Collection에서 대상 어플리케이션에 대한 트래픽을 수집하며, Wireshark를 사용하여 pcap 파일을 수집한다. 이후에 진행되는 과정에 따라 두 가지(i.e. 규칙 생성, 행위 탐지)로 나누어 수집된다. 행위 탐지에는 단순히 사용자 행위를 탐지하는 것이 아니라 어느 시점에 어느 호스트에서 어떤 정보를 통해 어떤 행동이 탐지되는지를 고려해야하기 때문에 트래픽 수집 단계에서 각 행위가 수행된 시간과 행위를 수행한 호스트 IP가 함께 기록되며, 로그 파일 형태로 저장한다. 즉, 규칙 생성 단계에서는 기록된 정보와 수집된 규칙 생성용 트래픽을 활용하여 보다 정확한 규칙을 생성하고, 행위 탐지 단계에서는 기록된 정보와 수집된 행위 탐지용 트래픽을 활용하여 생성된 규칙을 검증한다.

Traffic Processing에서 패킷 형태의 트래픽 파일을 플로우 형태의 파일로 변환한다. 플로우는 수집 된 패킷 중에서 5-tuples(i.e. Source IP, Source Port, Protocol, Destination IP, Destination Port) 정보가 같은 패킷들의 집합으로 정의한다. 수집 된 트래픽 셋은 이후에 수행되는 과정에 상관없이 동일한 전처리 과정을 거친다.

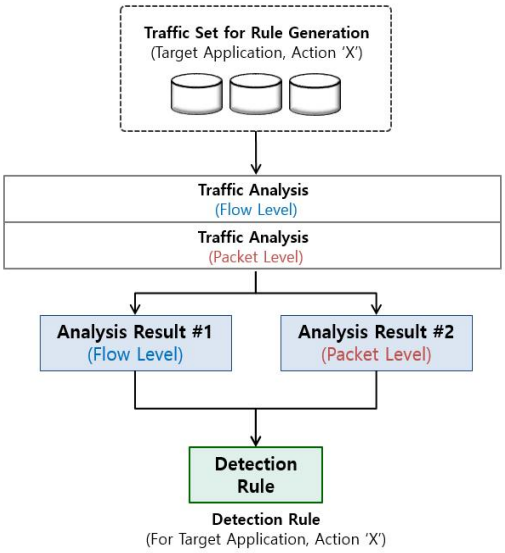


그림 2. 탐지 규칙 생성 과정
Fig.2. A Process of Detection Rule Generation

Rule Generation은 전처리 과정을 거친 트래픽에 대한 분석을 수행하고, 이를 기반으로 규칙을 생성하는 과정이며, 그림 2에 나타나있다. 탐지 규칙은 대상 어플리케이션의 행위들을 탐지하는 조건으로 각 행위 별로 공통적으로 발생하는 여러 가지 정보(Ex. 헤더, SNI 정보 등)로 구성되어 있으며, 행위 탐지 규칙에 대한 예시는 표 1에 나타나있다. 규칙 생성 과정은 대상 어플리케이션의 여러 가지 트래픽 데이터 셋을 활용하여 플로우, 패킷 레벨의 분석을 수행한다. 수행되는 트래픽 분석은 대상 어플리케이션의 개별 행위에 대해 여러 가지 트래픽 셋을 수집하고, 수집된 트래픽 셋을 확인하여 공통으로 발생하는 플로우 혹은 패킷을 찾는 것을 목표로 한다. 하지만 수행되는 분석은 수작업으로 공통 플로우, 패킷을 찾기 때문에 여러 가지 문제점을 발생시킨다. 따라서 추후에 기존 분석 방법의 문제점을 해결하기 위한 방법을 연구 할 예정이다. 수행 후 통해 도출되는 각각의 트래픽 셋 별 분석 결과를 취합하고, 공통적으로 발생하는 정보를 추출하여 탐지 규칙으로 정의한다.

생성된 행위 탐지 규칙은 크게 두 가지로 나

누어져있으며, 탐지 할 대상 어플리케이션과 행위로 이루어진 탐지 대상과 행위에서 공통으로 도출되는 트래픽 정보로 이루어진 탐지 규칙으로 구성되어 있다. 또한 탐지 규칙은 트래픽 형태에 따라 플로우, 패킷 기반 정보와 현재 탐지된 상태에 따라 선행 탐지 되어야 하는 행위를 나타내는 상태 정보로 구성된다. 어플리케이션 정보는 탐지 할 어플리케이션 정보를 뜻하며, 탐지 행위는 탐지 할 행위 정보를 나타낸다. 예를 들어 Microsoft Office 365의 어플리케이션 시작에 대한 규칙일 경우, 어플리케이션 정보는 Microsoft Office 365, 탐지 행위는 어플리케이션 시작으로 정의한다.

표 1. 행위 탐지 규칙 예시
Table 1. An Example of Action Detection Rule

규칙		정보 및 설명	예시
탐지 대상	어플리케이션 규칙	어플리케이션 정보	Adobe Creative Cloud
		탐지 행위	Login
탐지 규칙	플로우 기반 정보	클라이언트 IP	Any
		클라이언트 포트	Any
		프로토콜	TCP
		서버 IP	Any
		서버 포트	443
	패킷 기반 정보	확인 할 패킷 번호 (CPN)	4
		패킷 방향	CS (Client to Server)
		SNI	www.adobe.com
	상태 정보	선행되어야 할 행위 정보	Application Start

규칙 내 플로우 기반 정보는 각 행위 별로 공통적으로 도출된 플로우 정보이며, 클라이언트는 행위를 수행한 호스트, 서버는 호스트와 통신 한 목적지를 나타낸다. 특정 어플리케이션에서 행위 수행 할 때 고정된 플로우를 사용하여 통신 할 경우에 해당 플로우의 헤더 정보를 규칙으로 정의한다.

패킷 기반 정보는 각 행위 별로 공통적으로 도출되는 패킷 정보이며, 대부분의 어플리케이션에서 사용하는 TCP 기반의 TLS/SSL 패킷 페이로드는 암호화가 되어있기 때문에 암호화가 되어있지 않은 Handshake 패킷의 정보를 확인한다. 규칙 내 패킷 기반 정보 중 확인 할 패킷 번호는(Check Packet Number, CPN) 몇 번째의 패킷을 확인 하는지에 대한 정보이며, 패킷 방향은 확인 할 패킷의 통신 방향을 나타내며, 통신 방향은 CS(Client to Server)과 SC(Server to Client)으로 구분된다. SNI(Server Name Indication)는 TLS의 Handshake 과정 초기에 클라이언트가 어느 호스트명에 접속하려는지 서버에 알리는 역할을 하며, 문자열 값으로 구성되어 있다.

상태 정보는 탐지 할 행위가 수행되기 전에 어떠한 다른 행위가 선행되어야 하는지를 나타낸다. SaaS 어플리케이션에 대한 사용자 행위는 개별적으로 실행되지 않고 행위에 따라 순차적으로 실행되며, 다른 행위와 연관성을 가지기 때문에 현재 상태에 따라 탐지되는 정보가 달라 질 수 있다. 예를 들어, 4 가지의 행위 별로 어플리케이션 시작은 “None”, 로그인은 “어플리케이션 시작”, 로그아웃은 “로그인”, 어플리케이션 종료는 “어플리케이션 시작”이 선행되어야 하는 행위이다. 상태 정보는 오탐지를 줄이기 위해 정의하는 정보로, 선행된 행위를 확인하여 불필요한 오탐지 및 중복 탐지를 줄이는 역할을 한다.

생성된 규칙은 대상 어플리케이션의 하나의 행위에 대한 탐지 규칙으로, 최종적으로 하나의 어플리케이션에서 4 가지의 행위를 대상으로 4 가지의 개별 행위 탐지 규칙이 도출된다. 이후, 생성된 4 가지의 개별 행위 탐지 규칙은 대상 어플리케이션에 대한 하나의 탐지 규칙으로 취합된다.

Action Detection은 생성된 하나의 어플리케이션 행위 탐지 규칙과 사전에 수집된 행위 탐지용 트래픽 셋을 입력으로 사용자의 행위를 탐지하는 과정이다. 탐지 결과는 정탐지, 오탐지, 미탐지로 분류되며, 호스트 IP, 어플리케이션 정보, 행위 정보, 행위 수행 시점의 정보가 모두 일치 할 경우 정탐지로 판단하며, 하나의 정보라도 틀릴 경우 오탐지, 탐지하지 못할 경우 미탐지로 판단한다.

탐지 결과를 판단하는 기준은 사전에 트래픽 수집 단계에서 저장된 로그 파일을 활용한다. 세 가지의 탐지 결과를 활용하여 Recall, Precision, F1-measure를 계산하며, 수식은 아래의 수식 (1), (2), (3)에 나타나있다.

$$Recall = \frac{TP}{TP+FP} \quad (1)$$

$$\mathrm{Precision} = \frac{TP}{TP+FN} \quad (2)$$

$$F1-measure = \frac{2 \times \mathrm{Precision} \times Recall}{\mathrm{Precisio} + Recall} \quad (3)$$

도출된 탐지 결과에서 F1-measure 값이 특정 값 α보다 낮을 경우 잘못 생성된 규칙으로 판단하여 폐기하고 다른 정보를 통해 규칙 생성 과정을 반복한다. α 값은 임의로 설정하는 값으로, 본 연구에서 수행된 실험에서는 α 값을 0.95로 설정하였다.

Ⅳ. 실험 및 평가

제안하는 방법을 검증하기 위해 생성된 규칙을 활용한 행위 탐지 실험을 수행한다. 실험에는 대상 어플리케이션은 Adobe Creative Cloud를 사용하며, 사전에 언급한대로 어플리케이션 시작, 로그인, 로그아웃, 어플리케이션 종료의 4 가지 행위를 탐지 행위로 정의한다.

표 2. 트래픽 데이터 셋 정보
Table 2. An Information of Traffic Data Set

Purpose	App.	Trace	Traffic Information	
			Flow	Packet
Rule Generation	Adobe Creative Cloud	#1	985	8,022
		#2	1,021	8,258
		#3	1,125	10,552
		#4	845	7,125
		#5	625	5,255
		#6	1,225	13,758
		#7	328	6,255
		#8	512	5,220
		#9	524	8,802
		#10	425	7,775
Action Detection		#1	3,028	50,952
		#2	2,425	92,022
		#3	2,012	48,125
		#4	1,995	80,122
		#5	2,452	100,255

제안하는 방법의 타당성을 검증하기 위해 본 논문의 선행 연구인 시그니처 기반의 행위 탐지 방법[11]과 비교 실험을 수행한다. 시그니처를 활용한 행위 탐지 방법은 대상 어플리케이션에 대한 패킷 기반의 분석을 수행하고, 도출된 SNI 정보를 활용하여 행위를 탐지하는 방법이다.

실험에 사용한 트래픽 셋에 대한 정보는 표 2에 나타나있으며, Wireshark를 활용하여 트래픽 셋을 수집하였다. 총 15가지 Trace의 트래픽 셋을 수집하였으며, 규칙 생성을 위한 10 Trace와 행위 탐지를 위한 5 Trace로 구분된다. 규칙 생성 트래픽 셋에서는 Trace 별로 4 가지의 개별 행위를 1회 수행하였으며, 행위 탐지 트래픽 셋에서는 Trace 별로 4 가지의 개별 행위를 5회 수행하였다. 실험 결과는 행위 탐지 트래픽 셋을 대상으로 각 Trace 별로 전체 20회(4 가지 행위, 5회) 수행된 행위를 대상으로 정탐지(TP), 오탐지(FP), 미탐지(FN)을 확인한다.

실험 결과는 표 3에 나타나있으며, 각 Trace 별로 전체 20회의 행위 중 정탐지, 오탐지, 미탐지에 따라 계산한 Recall, Precision, F1-measure 값과 실험에 사용한 5 가지 Trace 결과 값의 평균을 기술하였다.

실험 결과로는 선행 연구에서 약 70~100%의 Recall과 70~89%의 Precision이 도출되었으며, 평균으로 약 78%의 Recall, Precision, F1-measure 값이 나타난다. 제안하는 방법은 약 88~100%의 Recall과 80~100%의 Precision이 도출되었으며, 평균 92%의 Recall, Precision, F1-measure 값이 나타난다.

표 3. 행위 탐지 결과
Table 3. Result of Action Detection

Method	Trace	Detection Result		
		Recall (%)	Precision (%)	F1-measure
Previous Method [11]	#1	87.50	77.77	82.35
	#2	78.95	71.42	74.99
	#3	78.75	89.69	83.86
	#4	100	50	66.67
	#5	82.50	71.42	76.56
	Avg.	85.54	72.06	76.89
Proposed Method	#1	100	80	88.89
	#2	94.44	89.47	91.89
	#3	95	100	97.43
	#4	88.89	94.44	91.58
	#5	94.44	89.47	91.89
	Avg.	94.55	90.68	92.34

제안하는 규칙 기반의 행위 탐지 방법은 시그니처 기반의 행위 탐지 방법에 비해 성능이 향상되었음을 알 수 있으며, 특히 Precision에서 상대적으로 차이가 많이 난다. 이는 시그니처 기반의 행위 탐지 방법은 SNI 정보만을 사용하기 때문에, 고정적인 SNI 정보를 사용하지 않거나 잘못된 SNI 정보를

시그니처로 정의 할 경우, 오탐지가 높게 발생 할 수 있다.

실제로 탐지 결과를 살펴보면 시그니처 기반의 분석 방법에서 SNI 정보를 시그니처로 잘못 정의되거나 실제로 특정 행위에서 고유하게 발생하는 정보가 아니라 다른 행위에서도 빈번하게 발생하는 SNI 정보일 경우가 다수 발생하여 오탐지 비율이 높게 나타난다.

Ⅴ. 결 론

본 논문에서는 사용자 행위 탐지 연구에 대해 소개하고, 행위 탐지 연구의 필요성에 대해 설명한다. 또한, 관련 연구에 행위 탐지 연구에 대한 기존의 여러 연구와, 본 연구의 선행 연구에 대해 소개하고, 선행 연구에 대한 문제점에 대해 언급한다.

본 논문에서는 관련 연구에서 언급한 선행 연구의 문제점을 해결하기 위해 규칙 기반의 행위 탐지 시스템을 제안한다. 제안하는 시스템은 규칙 생성과 행위 탐지의 두 가지 모듈로 구성되어있으며, 본론에서 각 모듈의 세부 구조에 대해 설명하였다.

제안하는 방법의 타당성을 검증하기 위해 선행 연구와 동일한 트래픽 셋을 활용하여 비교 실험을 수행 하였다. Adobe Creative Cloud를 대상으로 10 가지의 규칙 생성 트래픽 셋과 5 가지의 행위 탐지 트래픽 셋을 수집하였으며, 5 가지의 행위 탐지 트래픽 셋을 대상으로 성능 비교 실험을 수행하였다. 실험 결과로 제안하는 규칙 기반의 행위 탐지 방법이 전반적으로 더 높은 성능을 보이며, 특히 선행 연구와 제안하는 방법은 Precision에서 상대적으로 크게 차이가 났다. 이는 앞서 언급한 선행 연구의 문제점으로 오탐지 비율이 높게 나타나는 것으로 판단된다.

하지만 제안하는 방법에서 규칙 생성 할 때, 수행되는 트래픽 분석, 공통 특징 추출, 규칙 생성 등의 과정이 수작업으로 수행되기 때문에 많은 시간과 노력이 소모된다. 또한, 수작업으로 트래픽 분석 및 공통 특징 추출을 하기 때문에 선행 연구의 문제점에서 제시한 부정확한 정보를 규칙으로 잘못 정의하는 경우도 발생하였다. 또한, SNI 정보에 대한 의존도가 너무 높기 때문에, SNI 암호화와 같은 상황이 특정 어플리케이션에서 발생 할 경우에 탐지 정확도가 크게 낮아질 것으로 예상된다.

따라서 향후 연구로는 수동적 규칙 생성 방법을 개선하여 규칙 생성 시 수행되는 각각의 과정들을 일련의 과정으로 정리하고, 수작업으로 수행되는 분석 및 추출 과정을 자동으로 수행 될 수 있는 자동 규칙 생성 방법을 연구 할 예정이다. 또한 SNI 정보에 대한 의존도를 낮추기 위해 트래픽의 통계적 정보와 같은 다른 트래픽 특성을 함께 고려하여 규칙을 생성하는 방안을 연구 할 예정이다. 그리고 보다 다양한 어플리케이션을 대상으로 다른 방법론과 비교 실험을 통해 본 연구의 타당성을 추가로 검증할 예정이다.

References

[1] A. Callado, C. Kamienski, G. Szabo, B. Gero, J. Kelner, S. Fernandes, et al., “A Survey on Internet Traffic Identification,” IEEE Communications Surveys and Tutorials, vol. 11, 2009, pp. 37-52,

[2] A. Dainotti, A. Pescape and K. Claffy, “Issues and Future Directions in Traffic Classification,” Network IEEE, Vol. 26, no. 1, 2012, pp. 35-40.

[3] C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, and M. Rajarajan, “A Survey of Intrusion Detection Techniques in Cloud,” J. Network Computer Applications, vol. 36, no. 1, 2013, pp. 42 – 57.

[4] T. T. T. Nguyen and G. Armitage, “A Survey of Techniques for Internet Traffic Classification using Machine Learning,” IEEE Communications Surveys and Tutorials, Vol. 10, 2008, pp. 56-76.

[5] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, “Malware Traffic Classification using Convolutional Neural Network for Representation Learning,” in Proc. of 2017 International Conference on Information Networking (ICOIN), IEEE, Jan, 2017, pp. 712 – 717.

[6] M. Conti, L. V. Mancini, R. Spolaor and N. V. Verde, ″Analyzing Android Encrypted Network Traffic to Identify User Actions,″ in IEEE Transactions on Information Forensics and Security, vol. 11, no. 1, Jan. 2016, pp. 114-125.

[7] A. Shahraki, M. Abbasi, A. Taherkordi and A. D. Jurcut, ″Active Learning for Network Traffic Classification: A Technical Study,″ in

IEEE Transactions on Cognitive Communications and Networking, vol. 8, no. 1, March 2022, pp. 422-439.

[8] K. Park and H. Kim. "Encryption Is Not Enough: Inferring user activities on Kakaotalk with traffic analysis" International Workshop on Information Security Applications (WISA), 2015, pp. 254-265, Springer, Cham

[9] C. Hou, J. Shi, C. Kang, Z. Cao and X. Gang, "Classifying User Activities in the Encrypted WeChat Traffic," 2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC), 2018, pp. 1-8.

[10] H. Wu, Q. Wu, G. Cheng and S. Guo, "Instagram User Behavior Identification Based on Multidimensional Features," IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2020, pp. 1111-1116,

[11] 이민성, 최정우, 권윤주, 박지태 "페이로드 시그니처를 이용한 Adobe 소프트웨어 사용자 행위 탐지". 2021년도 통신망운용관리 학술대회 (KNOM 2021), 2021, pp. 24-25.

박 지 태 (Jee-Tae Park)

2017년 : 고려대학교 컴퓨터정보학과 학사

2017년 - 현재 고려대학교 컴퓨터정보학과 석박사통합과정

<관심분야> 네트워크 관리 및 보안, 트래픽 모니터링 및 분석,

백 의 준 (Ui-Jun Baek)

2018년 : 고려대학교 컴퓨터정보학과 학사

2018년 - 현재 고려대학교 컴퓨터정보학과 석박사통합과정

<관심분야> 네트워크 관리 및 보안, 트래픽 모니터링 및 분석

신 창 의 (Chang-Yui Shin)

2003년 : 육군사관학교 운영분석학과 학사

2007년 : 고려대학교 전자컴퓨터공학과 석사

2022년~현재 : 고려대학교 컴퓨터정보학과 박사과정

<관심분야> 네트워크 관리 및 보안, 트래픽 모니터링 분석

최 정 우 (Jung-Woo Choi)

2022년 : 고려대학교 컴퓨터정보학과 학사

2022년~현재 : 고려대학교 컴퓨터정보학과 석사과정

<관심분야> 네트워크 관리 및 보안, 트래픽 모니터링 및 분석

김 명 섭 (Myung-Sup Kim)

1998년 : 포항공과대학교 전자계산학과 학사

2000년 : 포항공과대학교 전자계산학과 석사

2004년 : 포항공과대학교 전자계산학과 박사

2006년 : Dept. of ECS, Univ of Toronto Canada

2006년~현재 : 고려대학교 컴퓨터정보학과 교수

<관심분야> 네트워크 관리 및 보안, 트래픽 모니터링 및 분석, 멀티미디어 네트워크

한국통신학회 통신망운용관리연구회
KNOM Review Vol.25/ No.2

ISSN : 2287-1543(Online)
2022년 12월 31일 인쇄
2022년 12월 31일 발행

발행인/ 석우진
편집인/ 김명섭 편집위원장
발행처/ 한국통신학회 통신망운용관리연구회
서울시 서초구 서초동 1330-8 현대기림오피스텔 1504동 6호
전 화 : 02-3453-5555
홈페이지 : www.knom.or.kr/knom-review/